ONREUR Report

9-1-C

The 2nd International Conference on Vector and Parallel Computing

J.F. Blackburn

17 January 1989

Office of Naval Research European Office

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) 9-1-C	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Office of Naval Research European Office 6b OFFICE SYMBOL (If applicable) (ONREUR)	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State, and ZIP Code) Box 39 FPO, NY 09510-0700	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

The 2nd International Conference on Vector and Parallel Computing

12 PERSONAL AUTHOR(S)
J.F. Blackburn

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Conference	FROM _____ TO _____	17 January 1989	42

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
			Vector Computing; Parallel Processing; Supercomputers; Parallel Programming; Vectorization; Parallelization; (KT)

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Summaries of the presentations by invited speakers to this conference, held in Bergen, Norway, are given along with the authors' abstracts of the contributed and student scholarship papers. In all, summaries of 16 papers and the abstracts of 91 other papers are included.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL C.J. Fox	22b TELEPHONE (Include Area Code) (44-1) 409-4340 22c OFFICE SYMBOL 310

☆ U.S. Government Printing Office: 1986—607-044

Contents

Student Scholarship Winners

THE SECOND INTERNATIONAL CONFERENCE ON VECTOR AND PARALLEL COMPUTING

Introduction

This 5-day conference was organized principally by the IBM Bergen Scientific Center with the participation of the Society for Industrial and Applied Mathematics and the Association for Computing Machinery. Held in Bergen, Norway, the conference was attended by 450 participants coming mainly from Western Europe and the United States.

The program included invited speakers, contributed papers, and student scholarship papers. My report includes summaries of the lectures given by the invited speakers based on notes taken during the conference and the authors' abstracts of the contributed and student scholarship papers. These were given in parallel sessions, only half of which I could attend.

Invited Speakers' Presentations

"The Grand Challenge of Supercomputing"

Alan Weis, keynote speaker, IBM Data Systems Division, US, Vice President of Engineering and Scientific Computing.

There is an ever-changing role for supercomputers in the modern scientific community. One important challenge is that of accelerating the absorption of supercomputers into the broader community of busy users – as contrasted with computer specialists.

The predominant tools of investigation in experimental science of the past have become too costly to use. The need is increasing for more numerical approaches to solving these problems. Hence there is a growing need for supercomputers in such areas as aircraft design, weather forecasting, and exploration in the oil industry.

Today the mainstream supercomputer, viewed by the busy user, is expensive, hard to use, experimental, and limited to highly specialized users. The busy user of computers simply wants to solve problems in science and mathematics without the need to become a super specialist in computer architecture. The need for the mainstream user of the future is a system which is cost effective, reliable, easy to use, rich in software for applications, and robust in the data management.

Why are users moving toward the supercomputer? It is becoming widely recognized by government, industry, and universities as a useful and more and more available tool for solving problems requiring lots of data and very fast computation. The initiatives of the NSF in the US along with various European initiatives are helping to bring this about. New applications requiring extensive computation like the use of fractals for various problems in physics and engineering and the use of finite elements in fuselage design in the aeronautics industry are bringing about more and more demand for supercomputers. Also, manufacturers are producing more powerful and more robust systems than ever before. And this technology is being put into the hands of users.

The technology transfer of the supercomputer is through the academic and research scientists, industrial research scientist, and the industrial and commercial end user. This move to supercomputers is happening now because the environment for their use is better understood and the requirements for the applications are being better addressed. The necessary components for further extending the use of supercomputers are technological advances and architectural developments.

The relevant technological advances in logic and memory are related to miniaturization, price reduction, tools, and facilities. Significant developments relating to miniaturization include the scanning tunneling microscope, better understanding of materials at the atomic level, and understanding the silicon surface. Improved processing tools include E-beam and x-ray lithography. Another important development is that of advanced facilities like super-clean rooms in which to produce components.

Important advances in field effect transistors will lead to one million transistors per chip and, of perhaps equal importance, are the developments in packaging, which requires an understanding in depth of materials. Precision in disk storage devices now permits read/write heads to be 10 millionths of an inch above the disk surface. This disk surface must, of course, be totally flat. Optical storage has been developed to the point where laser-written pits can be at a spacing of five microns.

Developments in system architecture include higher performance in both scaler and vector processors, more parallelism for both general purpose and special purpose computers, larger memories, and much higher bandwidth for communication with input and output subsystems.

More mature system software is now available to permit interactive computing with systems having very large virtual memories. Management control systems are greatly improved and computer technology for vectorization and parallelism is now becoming available. Data management systems now offer greater reliability and faster access. Network connectivity is evolving toward

Dr. Blackburn is the London representative for the Commerce Department for industrial assessment in computer science and telecommunications.

standard protocols like OSI. Workstations are more thoroughly integrated into systems allowing cooperative processing.

This year, using X-windows on a vector computer scientists were able to observe and steer processes and "fly" over a silicon chip for more information on chemical bonds through changing the current applied to the material.

Network access and interconnection is featuring high bandwidth, management for service through programs and data libraries, and migration to OSI standards. Current networks in the US, Europe, and Japan will shortly feature speeds up to 45 Mb/sec. They allow expansion as in the US NSF network connection to Europe and Japan.

Present-day networks do not do a very good job of network management. NSF will concentrate on network management and the use of program libraries to help in cross-discipline use.

Traditionally the architecture of a computer system influenced the algorithms used with the system in applications. The trend now is toward the algorithm influencing the architecture.

Among the grand challenges remaining are: getting the supercomputer to be more widely used by the busy user, the integration of CAD/CAE/CAM, and use in computational chemistry. Government, universities, and industry need to work together to apply the required interdisciplinary resources to meet these challenges. And they must work together to insure that educational needs are adequately met.

"Supercomputing as a Tool for Product Development"

Alan Erisman, Boeing Computer Services.

As Richard Hamming said in 1962, "The purpose of computing is insight, not numbers." However, in much of today's computing the numbers are very important. The computing requirements of such areas have data processing characteristics that have long differed from scientific computing requirements where data is analyzed as opposed to processed. FORTRAN, supercomputers, minicomputers, and workstations dominate today's scientific computing; fourth-generation languages, mainframes, and personal computers dominate data processing.

Because of this background, supercomputers are associated with scientific analysis; the use of today's vector and parallel computers requires that code be adapted to the architectural environment to use the central processing units effectively. The proliferation of so-called near-supercomputers has made vector and parallel architecture available to a broader group of engineers and scientists.

The use of these analyses results in industry requires that they be accessible to the other part of the computing world where products are made. Manufacturing considerations place design constraints on the products to be built, and this changes the models which must be analyzed. In order to produce a product at low cost, the analysis alternatives need to include pricing data.

The separate worlds of scientific computing and data processing will have to come together. In sophisticated analyses made possible by the supercomputer users know too little about CAD/CAM, and vice versa.

Supercomputers are powerful engines which provide opportunities to solve problems previously intractable. Computer systems provide powerful computing hardware operating systems, compilers, and languages. Scientific computing provides algorithms, applications programs, and a computing environment with, for example, data management.

Some successes due to supercomputers in science have occurred in understanding molecular structure and fluid flow under various conditions. Successes in products include the design of the Boeing 737-3000, the Ford Taurus, and enhanced oil recovery. The benefits relate to product performance rather than to technical accomplishments. The supercomputer is more than a research and development tool.

The supercomputer impact on the design of the Boeing 737-3000 had to do with the engine placement in relation to the total aerodynamics of the entire airplane. In products the supercomputer may benefit a design by showing that a small change in design may produce a large performance difference. This may result in significant economic benefits.

We need new supercomputer models for dealing directly with product development. There is a close link between research and product development, and also a greater need for the supercomputer to be used more frequently by the so-called busy user who doesn't have time to become a computer expert. We need to deal with all the available or obtainable data.

There is a potential for computers to do more in product design. The requirement is for ever-more-powerful supercomputers. This presents an opportunity and a challenge to the computer manufacturer. New modeling approaches are possible through better analysis and design optimization.

Parallelism in computers poses a tough challenge. There is very little software experience with parallelism.

Libraries have an important role in supercomputing. The performance of a vector computer is closely tied up with the application – e.g., dependent on the amount of natural parallelism in the application program. Highly tuned libraries of application programs can have an important impact on performance. In a parallel computer not only are the architecture and the algorithm important to performance but also the application.

Modeling involves design optimization – integrated analysis including structural design, control systems, thermal design, and aerodynamics design. It may also include artificial intelligence and symbolic computing. Manufacturability issues must be considered in modeling and de-

sign. The total analysis process needs to be integrated into one application.

At present the supercomputer is involved in performance analysis but not in CAD/CAM. Thus, the performance analysis is not carried through as it should be. There is a serious need for proper integration of the whole process.

Integration in the computer sense involves the supercomputer, the work stations, graphics capability data management, high-bandwidth communication, a common operating system, and artificial intelligence tools. The user at his terminal is the foreground of a properly integrated system, and all else is background.

In summary, we need more powerful supercomputers, better modeling and algorithm development, integrated computer systems, better data management, and company organization.

"On the Suprenum System"

Ulrich Trottenberg, GMD, West Germany.

The Suprenum is a supercomputer for numerical applications. The hardware is a highly parallel MIMD architecture in which nodes of processors have vector units and local memory. It was not considered to be a good approach to design and optimize for a particular algorithm, to design sequentially for a conventional computer, or to design special purpose computation for certain large-scale applications on the basis of old fashioned algorithms. The idea was to design a system for general large-scale applications.

Suprenum 1 is scheduled for completion at the end of 1989 with a planned performance of 4 Gflops. Suprenum 2 is a research project but will be a product at a later date.

The node of a Suprenum will consist of 16 worker processors (MC 68020) each with a floating point vector unit (Weitek) rated at 16 Mflops and local memory. A cluster will consist of 16 nodes for a total of 256 processors. A high-performance system would consist of 4x4 clusters extendable to 16x16 clusters.

Although a 256-processor system would have a theoretical maximum performance of 4 Gflops, a more realistic actual performance is likely to be 1-2 Gflops.

The system is architecturally a compromise between full connectivity of all processors in the system and strictly local connectivity. The architecture is a two-level bus-coupled architecture.

An abstract Suprenum machine exists which allows hardware-independent programing. There is also a Suprenum FORTRAN and a concurrent Modula I-2. The Suprenum FORTRAN is an extended FORTRAN 77 with process handling, message passing, and array operation capability. Also, the concurrent Modula-2 is an extension of the original Modula-2. There will also be a communication library for Grid applications. An important tool is a dynamic map which gives a picture of all processors at a given instant.

A basic numerical library of applications is planned which will cover linear algebra, multigrid solvers for partial differential equations, and an ordinary differential equations package. Another package will provide full potential equation solutions for subsonic and transonic flow.

There was an 8-node version of Suprenum running as of April 1987. A 32-node version is expected to be operational in the first quarter of 1989 and will be demonstrated at the Hannover Fair, 5 April 1989. The system is to be manufactured for marketing by 1990.

"IBM Supercomputing Trends and Directions"

Alec Grimison, IBM.

The base for supercomputers in IBM is the 3090-600E to which can be added up to six vector processors. The 3090 is an outstanding scalar processor. Its vector capacity is optimized for applications that are 60-70 percent vector. The system has a large memory, and expanded ranging from 256 Mb to 2 Gb. It has an excellent aggregation of input/output equipment and is a very high performance system.

Its virtual store-extended architecture software MVS/ESA has 16 Tb addressable. It also supports VM/XA and AIX/370. The VS FORTRAN vectorizing compiler and parallel FORTRAN for single-job turnaround are available on the 3090-600E. There are more than 40 application packages available with the system.

The critical elements in IBM's supercomputer strategy are:

- A balance between scalar, vector, and parallel processing
- Large memories and input/output capability to match the system's computing power
- Compatibility with existing systems
- High use by balanced capability between the need for throughput and turnaround
- Moderate parallelism with shared global memory.

The strategy includes firm coupling:

- Connection between IBM 3090 complexes to provide very-large-scale scientific computing
- Careful balancing of hardware, software, and systems requirement
- High performance.

The current hardware allows multiple 4.5-Mb/s channels. The need for higher data rate is foreseen and being explored. IBM is an active participant in the standards group, ANSI, which is drafting standards for high-speed connection interfaces for up to 100 Mb/s.

Multiple 3090 complexes do not have shared memory in the S/370 sense. Parallel FORTRAN currently has shared and private common areas. Extensions to parallel FORTRAN will make designated common blocks of

memory appear globally shared. The effect on performance is being explored.

An applications set of programs is being analyzed to handle situations between the extremes of obviously parallel and nonparallel subsets. Also the balance between high-communications with compute-ratio is under study. There are ongoing studies at Cornell University and IBM Yorktown Heights Research Center.

A hierarchical approach to processing will allow decomposing a problem among complexes with computation time in excess of intercluster communications.

The subject of visualization in scientific computing needs to be defined, and the extent of need should be determined. It may be defined as interactive display and processing of data from an ongoing supercomputer. The need is to steer simulations in computationally real time. A further need is for higher resolution color graphics, and much higher communication bandwidth.

Further along we will evolve very high-end extended systems architecture ESA/370 for large-scale computer systems. Such a system will build on the commercial hardware/software base which will be extended. Super scalar performance characteristics will be exploited. A system balance will be maintained between vector, scalar, parallelism, memory, and input/output.

IBM is wedded to the future of parallelism. We shall optimize to high vector/parallel content as applications evolve. We shall optimize to user demands such as memory bandwidth for technical computing. The ESA/370 hardware and software technology will be driven to its limit. There will be some fallout for use in the commercial line of computers.

IBM intends to be a major participant in providing supercomputer solutions. We have the appropriate base to allow for rapid enhancement. We will continue to emphasize the balanced systems approach.

"Parallel Logic Programing"

David Warren, Manchester University, UK.

Japan's Fifth-Generation project has helped to highlight logic programing as a unifying framework on which to build advanced computer systems, particularly for nonnumeric applications. One of the key features of logic programing as a framework is that it provides a powerful model of computation which lends itself to parallel implementation.

There are two main kinds of parallelism in logic programs: or-parallelism and and-parallelism. Or-parallelism enables alternative solutions to a query to be found in parallel. And-parallelism enables different steps toward a single solution to a query to be performed in parallel.

The performance of computers running logic programs is measured in logic inferences per second (LIPS). Manchester with the support of the Science and Engineering Research Council (SERC) is working with MCC, Austin, Texas, toward a powerful system.

The project is motivated by the cost effectiveness of using multiprocessors and the difficulties in programing for them. There will be wide acceptance of multiprocessing only when the systems are regarded by programers as a "black box."

Prolog was chosen because it is adequate for real applications; it is widely known and used and is a well adapted technology. Prolog can be considered a generalization of a functional language like pure LISP.

The Japanese institute ICOT is using dependent and-parallelism in languages like Prolog and Concurrent Prolog. We at Manchester are working with or-parallelism, as in the SRI model. Our aim is to run real applications faster without changing the program.

Our system is called the Aurora system and is constituted of Sicstus Prolog plus the SRI model + a scheduler. Aurora is a full Prolog system with good speed and speed ups.

In the future we expect to explore new applications and the scheduling of speculative work and to incorporate dependent and-parallelism in an Andorra model which will combine and- and or-parallelism.

We want processors to share data rather than memory. Each datum will be identified by a virtual address. Virtual addresses can be mapped quite flexibly onto physical addresses. There may be multiple copies of a particular datum and the physical location of a datum will be transparent to the user. Data will simply migrate to where it is needed.

A communications controller will handle local and nonlocal memory accesses. Local communication will allow reading of a local datum or write an unshared datum. Nonlocal communication will allow reading a nonlocal datum, broadcast to nearest copy and mark as required.

A data diffusion machine is characterized by shared virtual memory but not shared physical memory. It is scalable and data migrates automatically to minimize remote access.

"Domain Decomposition Algorithms and Applications to Fluid Dynamics"

Tony Chan, UCLA.

Domain decomposition is a class of methods for solving mathematical physics problems by decomposing the physical domain into smaller subdomains and obtaining the solution by solving smaller problems on these subdomains. Motivation for this approach may be:

- The ability to use different mathematical models and approximation methods in different subdomains
- Use of fast, direct methods in subdomain
- Memory limitations of the computer
- Suitability for implementation on parallel computers.

Applications can be found in many areas of scientific computing, such as computational fluid dynamics and

structural mechanics. The key ingredient in many of these methods is the system of equations governing the variables on the interfaces between the subdomains which is often solved by preconditioned iterative methods.

One question that arises is whether or not to have overlapping subdomains. In the view of Professor Chan it doesn't seem to make much difference.

In solving partial differential equations on a domain using either finite difference of finite elements the domain is decomposed into subdomains, on each of which the solution is simpler than on the entire domain. The subproblems are solved on the subdomains and these solutions are pieced together to arrive at a global solution on the entire domain.

The piecing together involves solving the system of equations governing the variables on the interfaces between the subdomains. One approach is to estimate the solution on the interior boundary and carry out successive iterations until the equations are satisfied to an acceptable level. Often a preconditioner is used such as the Preconditioner Conjugate Gradient.

"Domain Decomposition Methods for Parallel Computer"

Gerard Meurant, Centre d' Etudes de Limeil-Valenton, France

Domain Decomposition methods were originally developed to solve large problems on computers with small memory or to decompose problems on complex geometries, allowing fast methods to be used on the subdomains. Today, these methods have become interesting for use with parallel computers, mainly of the MIMD type.

The oldest of the domain decomposition methods is the Schwarz Alternating Method with overlapping subdomains. Some new results both on convergence and applications of this method have recently been found. The other methods are mainly related to the Conjugate Gradient Method, as ways to derive efficient preconditioners. The methods are classified by the kind of problems they can solve or by the type of parallel computer to which they are best adapted.

Some methods rely on knowledge about the underlying partial differential equation, use direct solvers on the subdomains, and so are targeted to parallel computers with a very large number of processors. Some others are purely algebraic methods and use approximate solvers on the subdomains, and hence are more suitable for computers with small numbers of powerful vector processors.

The procedure in any case is to split the problem into pieces, solve the pieces in parallel on the subdomains, and then put the pieces together to get the global solution.

Domain decomposition methods differ in several ways:

- The method of partitioning may be with or without overlapping of subdomains and the subdomains may be stripes or boxes over the domain.
- The method of solution on the subdomain may be exact, approximate, or an exact solution of an approximate problem.
- The method of construction of the problem for the interfaces may be from the partial differential equation or algebraically from the matrix of coefficients.

Meurant's group has chosen a target architecture using stripes and a supercomputer with a few very powerful processors using shared memory. With the use of stripes as subdomains they will be looking for long vectors.

The Schwarz method of solution uses a block Gauss-Seidel method applied to the matrix of coefficients. A large number of iterations is required when there is little overlapping of subdomains but the number of iterations drops dramatically for large overlapping.

Other methods of solution are to use the Conjugate Gradient method to accelerate the convergence process on the interface, or to use the Block Jacobi method with overlapped subdomains.

The Domain Decomposition method is especially well suited to parallel supercomputers.

"Comparison of Super and Mini-Super Computers for Computational Fluid Dynamics Calculations"

Wolfgang Gentzsch, Fachhochschule Regensburg, West Germany.

A model benchmark has been developed to estimate the performance of supercomputers for engineering and scientific applications. It consists of four parts: special kernels, basic linear algebra routines, iterative solvers for systems of equations, and application programs.

By using about 100 kernels and basic linear algebra routines it is possible to:

- Test the capability and the limits of the vectorizing and parallelizing compiler
- Estimate the performance for basic operations depending on vector length.

Twenty-five variants of different linear algebraic systems solvers are used to:

- Study restructuring with respect to the special vector and parallel architecture
- Conclude basic vectorization and parallelization rules for numerical algorithms.

In addition, five production codes from plasma physics, Euler and Navier-Stokes Flow, Grid Generation, and Multigrid have been included to get better insight into more complicated constructs within more complex programs.

The solution of equations like the Navier-Stokes involving the conservation of mass, momentum, and energy involve a number of steps:

- Mesh generation
- Discretzation of the partial differential equations

- Determining a starting solution
- Modeling of turbulence
- Including stability and convergence
- Applying artificial viscosity

and may include

- The use of computer graphics.

Such problems involve a very large number of unknowns – as many as 30 to 40 at every grid point. Thus, a 30x30 grid would involve 36,000 unknowns.

The results of benchmarking of such problems depend on the machine architecture, the compiler used, and the algorithms. Thus, benchmarks can be misused, the choice (or omission) of optimal kernels can influence the average machine. Often decisions are made on benchmark results which are not entirely indicative of performance.

"Images of Matrices"

Cleve Moler, Ardent Computers, US.

That supercomputers have been proven effective for computation but have not yet been proven for visualization is a paradox. Mathematical visualization means the use of powerful graphics software and hardware to investigate mathematical computations.

The underlying tools for mathematical visualization include:

- Titan, Ardent's new graphics supercomputer with super parallel architecture with various levels of parallelism including pipelining and vector parallelism
- Software, consisting of four groups: an operating system; compilers for various languages; graphics (Ardent's Dynamic Object Rendering Environment [DORE]); and science software, e.g., Math Work's Matrix Laboratory (MATLAB).

DORE provides a connection between computing and graphics. It takes geometric components to produce realistic objects.

The classic MATLAB was developed by Moler at Argonne National Laboratory and Stanford University. The commercial version of MATLAB is written in the language C and is useful in two- and three-dimensional graphics. It is also extensible.

Examples of the use of MATLAB and DORE include:

- A dynamic portrait of a vibrating L-shaped membrane
- A view of matrix decomposition algorithms
- Surfaces defined by mapping of the complex plane
- Solutions of some model partial differential equations.

"Parallel Integration of Vision Models"

James Little, Artificial Intelligence Laboratory, MIT.

Computer vision has developed algorithms for several early vision processes – such as edge detection, stereopsis, motion, texture, and color – that give separate cues to the distance from the viewer of three-dimensional surfaces, their shape, and their material properties. Yet, and not surprisingly, biological vision systems still greatly outperform computer vision programs. It is increasingly clear that one of the keys to the reliability, flexibility, and robustness of biological vision systems is their ability to integrate the different visual cues. We have developed a technique to integrate different visual cues, and have implemented it with encouraging results on a parallel supercomputer.

Whereas it is reasonable that combining the evidence provided by multiple cues – for example, edge detection, stereo, and color – should provide a more reliable map of the surfaces than any single cue alone, it is not obvious how this integration can be accomplished. One of the most important constraints for recovering surface properties from each of the individual cues is that the physical processes underlying image formation, such as depth and orientation and reflectance of the surfaces, are typically smooth. Standard regularization (Poggio and Torre, 1984), on which many examples of early parallel vision algorithms are based, captures this smoothness property well.

The physical properties of surfaces, however, are smooth almost everywhere, but not at discontinuities. Reliable detection of discontinuities is critical for a vision system since discontinuities are often the most important locations in a scene. The idea is to couple different cues to the image data (especially intensity edges) through the discontinuities in the physical properties of the surfaces. The goal is, of course, to use information from several cues simultaneously to help refine the initial estimation of surface discontinuities, which are typically noisy and sparse.

How can this be done with an algorithm that is intrinsically parallel? We have chosen to use the machinery of Markov Random Fields (MRF's), initially suggested for image processing by Geman and Geman (1984). We have extended our previous work (Marroquin et al., 1987) to couple several of the early vision modules (depth, motion, texture, and color) to intensity edges in the image. This is a central point in our integration scheme: intensity edges guide the computation of discontinuities in the physical properties of the surface, thereby coupling surface depth, surface orientation, motion, texture, and color each to the image intensity data and to each other.

We have been using the MRF machinery with appropriate prior energies to integrate edge-intensity data with stereo, motion, and texture information on the MIT Vision Machine System. The system consists of a two-camera eye-head input device and a 16K Connection Machine. All the early vision algorithms – edge detec-

tion, stereo, motion, color, and texture – as well as the MRF algorithm, currently run on the Connection Machine several hundred times faster than on a conventional machine.

At the same time, our integration algorithm achieves a preliminary classification of the intensity edges in the image, in terms of their physical origin. Preliminary experiments suggest that recognition algorithms can use effectively the output of the integration scheme described here.

These highly parallel algorithms map quite naturally onto an architecture such as the Connection Machine, which consists of 16K simple 1-bit processors with local and global connection capabilities. These algorithms also map onto VLSI architectures of fully analog elements and mixed analog and digital components.

"Programing Parallel Vision Algorithms: A Dataflow Language Approach"

Linda Shapiro, University of Washington, Seattle.

Computer vision requires the processing of large volumes of data and requires parallel architectures and algorithms to be useful in real-time industrial applications. The INSIGHT dataflow language was designed to allow encoding of vision algorithms at all levels of the computer vision paradigm. INSIGHT programs, which are relational in nature, can be translated into a graph structure that represents an architecture for solving a particular vision problem or a configuration of a reconfigurable computational network.

Single-processor, general purpose computers cannot provide the computational power required for real-time or even reasonable-time vision tasks. At the image processing level, parallelism has been achieved to some extent by cellular array machines, pipeline architectures, and pyramids. In order to deal with more complex vision problems, including low-level, mid-level and high-level algorithms and to provide a greater computational resource, massively parallel cellular machines such as the Connection Machine and MIMD machines like the Butterfly have been built.

In addition, a new tri-level parallel architecture providing a large array of simple processors for image processing, a medium-sized array of more powerful processors for mid-level vision, and a small array of extremely powerful processors for high-level algorithms is being developed in conjunction with the DARPA Image Understanding Project. Since most of these new machines are intended for defense use, they are currently much more expensive than industry is willing to pay for machine vision. For this reason, reconfigurable architectures that have less processing elements, but can be reconfigured to solve a variety of problems are being proposed.

All of these machines need a language in which vision algorithms can be expressed. If the language reflects the architecture of the machine, then software sharing between installations with different machines will be impossible and much unnecessary effort will go into developing and redeveloping parallel algorithms.

A more desirable approach is to have a non-machine-dependent language that can express parallel algorithms in a generic way and can be translated to code that runs on a particular architecture or to a configuration of a reconfigurable architecture. This was the approach in the design of INSIGHT, a dataflow language for programing vision algorithms. INSIGHT can be used for expressing low-level mid-level and high-level vision algorithms, and INSIGHT programs can be translated to code that can run on a variety of architectures.

"Seismic Wave Propagation and Absorbing Boundary Conditions"

Johnny Peterson, Bergen Scientific Centre, IBM, Norway.

When computing solutions to the two-dimensional wave equation in unbounded domains using finite difference discretization, an artificial boundary is introduced. A boundary condition which absorbs all outward propagating waves must then be used. Also, the finite difference operator must be replaced at the boundary with an appropriate boundary operator.

Approximation to boundary operators are well known which work well for waves which are propagating towards the boundary near normal incidence. However, problems occur in cases where sources are far from the center of the model or if the velocity field is not homogeneous. In such cases results are contaminated with noise scattered back from the artificial boundaries.

A nonlinear least squares method is proposed for determining an absorbing boundary operator. The operator is chosen by demanding that waves traveling within a predetermined cone are alternated as much as possible. The problem is solved with a Monte Carlo-type minimization method. Storage requirements can be reduced by reducing the grid size. Results can be obtained from the vectorization of the method.

"Large-Scale Computing in Reservoir Simulation"

Richard Ewing, University of Wyoming.

The objective of reservoir simulation is to understand the complex chemical, physical, and fluid flow processes occurring in a petroleum reservoir sufficiently well to be able to optimize the recovery of hydrocarbon. For this, mathematical and computational models must be built capable of predicting the performance of the reservoir under various usable schemes. Many of the physical phenomena which govern enhanced recovery processes have very important local character. Therefore, the models used to simulate these processes must be capable of resolving these critical local features.

Mathematical models of enhanced recovery processes involve large coupled systems of nonlinear partial differential equations. In order to compare the results of these models with physical measurements to assess

their validity and to make decisions based on these models, the partial differential equations must be discretized and solved on computers. Field-scale hydrocarbon simulations normally involve reservoirs of large size. Uniforms gridding on the length scale of the local phenomena would involve systems of discrete equations of such size as to make solution on even the largest computers prohibitive. Therefore, local grid refinement capabilities and efficient solution processes are becoming more important in reservoir simulation as the enhanced recovery procedures being used become more complex, involving more localized phenomena in enormous problems.

Equations representing the miscible displacement of one incompressible fluid by another, completely miscible with the first are combined and lead to equations describing multiphase and multicomponent flow in porous media. These can be used to simulate various production strategies in an attempt to understand and optimize hydrocarbon recovery.

In miscible or multicomponent flow models, the connective, hyperbolic part of the equation is a linear function of the fluid velocity. The operator-splitting technique applied to a variational method leads to a symmetric bilinear form. A modified method of characteristics is used to treat the time stepping. The discretization methods used can be considered as the first step in a Newton linearization of the coupled nonlinear system. The method is designed to linearize and formally decouple the equations for a sequential solution process. However, in cases where the nonlinearities in the partial differential equations are strong this linearization process is not sufficiently accurate for the desired application. In such cases the full Newton-Raphson type of treatment can be used.

"ParaScope: A Parallel Programing Environment"

Ken Kennedy, Rice University.

Clearly, future generations of scientific supercomputers will employ multiple independent processors. What form of programing support software should be provided with such machines? Existing FORTRAN programs, written for sequential machines, are not well suited to parallel execution. If these programs are to run efficiently on a multiprocessing system, they must be decomposed into subproblems that can be executed in parallel.

Although there has been substantial progress in methods for automatic transformation of sequential programs to parallel form, there is little evidence that these methods will make it possible for the programer to be unconcerned about parallelism. We must therefore assume that parallel programs will be written by human programers in an explicit parallel notation.

Explicit parallel programing is a challenging activity fraught with opportunity for error. If programers are to be productive on the next generation of machines they will need powerful new tools to assist in the programing process. The ParaScope project at Rice University is planned to provide such tools in the context of an integrated programing environment.

ParaScope is based on a sophisticated environment for FORTRAN programing developed over the past 5 years at Rice. In addition to the usual tools, such as editors, compilers, and source-level debuggers, ParaScope will incorporate new tools specifically designed for parallel programing, including an editor that interactively reports potential sources of inadvertent data sharing between parallel processes, a compiler than analyzes the whole program to produce good parallel code, a debugger that attempts to execute parallel programs according to a schedule likely to recreate data sharing errors and performance visualization tools that help the users identify run-time bottlenecks in their programs. A central theme in the design of the system is the use of deep program analysis methods, developed for automatic transformation systems, in the programing and debugging tools.

"Current Directions and Future Possibilities in Computational Fluid Dynamics"

Anthony Jameson, Princeton University.

This paper covered a wealth of material, but the talk moved too fast for easy following. The speaker reviewed mathematical models suitable for different flight regimes, and current developments in the design of algorithms for their numerical simulation. Estimates of corresponding computational requirements of both speed and memory were included, and the impact of massively parallel architectures on future possibilities for numerical simulation of fluid flows was assessed.

The whole emphasis was on computational aerodynamics, which requires identification of the relevant physical phenomena and the formulation of appropriate models. In the solution of such problems there is a role for mathematics (including numerical analysis), computer science (including how to prove a program correct), aeronautical engineering (including what is the objective).

The specific objective is to calculate the flow pattern of the air past the aeroplane. This involves calculating the flow past the aeroplane in different flight regimes and requires interactive information. The flow pattern will involve geometric complexity and must take into account viscous effects.

The steps involved in the process include:

- The choice of a mathematical model (this choice ranges from a Laplace equation for ideal fluid flow to Navier-Stokes equations for complex cases)
- Analysis of the model chosen
- Derivation of a numerical approximation to the partial differential equations involved

- Writing of a program to solve the approximately equation
- Validation of the model and the program.

The Euler equations in aerodynamics fall between the Laplace equation and the Navier-Stokes in complexity. In aerodynamic design there is normally a tradeoff between complexity of the algorithm and the model. The chosen algorithm may be finite differences or finite elements. It may involve time marching or be steady state.

"Parallel Programing with Ada"

Jan Kok, Centrum Voor Wiskunden Informatica, the Netherlands.

The language Ada (ANSI/MIL-STD 1815 A, 1983) was primarily designed for the production of large portions of readable, modular, portable, and maintainable software for real-time applications. In the programing area concerned with this production the differences between machines, systems, languages and language implementations, experienced when transporting and maintaining programs, are a main cause of errors in programs.

In order to provide the means for obtaining more reliable software for these applications the US government launched a significant program in the 1970's with the result that both the specification was given of a portable environment for developing and running software, and a high-level language was defined with properties that should enhance the programing of reliable and maintainable software. This language, Ada, offers standard and readable language concepts for the structuring of large programs, for the specification of the relationship between different modules of a program, for data abstractions, and for programing distributed computing with clear tools for describing processes and the communication between these.

In this presentation the author focused on the language as an appropriate tool for the human user. High-level languages have the property that the step from algorithms (formulated with natural language or mathematical notation) to programs in those programing languages is small and can also be done in the reverse direction due to the readability of the code.

This property comes along with a high degree of abstraction away from particular hardware or system characteristics, which actually puts the burden of directly addressing the machine possibilities on the specific compilers. In particular for parallel programing, the intentionally standardized languages with clear and high-level features are rare. The necessity for high-level features is not generally accepted, and the suitability of possible expressive tools like those of Ada has not been extensively investigated. Presumably many believe that such tools are not possible with the expected diversity of parallel architectures.

With the following description of the Ada tools for programing parallel actions and of the possibilities to exploit parallel architectures the author intends to bring to a broader forum the issue of language tools for parallel programing. This may hopefully result in feedback for increasing the understanding about the applicability of these tools, and also for their improvement in Ada and in other scientific languages for which parallel programing tools are under development.

The author first reviewed the relevant Ada concepts that can be used for parallel programing, in particular the task concept and the related declarations and statements that can be exploited.

Next, the possibilities were discussed for the supposed and efficient mapping of Ada tasks into existing and imaginable multiprocessor architecture. He indicated some observed disadvantages of particular language constructs and reported experience gained in model exercises which can be useful for the solution of numerical problems as well.

Finally, he discussed the possibilities for implementing in Ada parallel methods and for developing new methods with the help of the readable Ada features, where this development so far has been handicapped by the lack of high-level language concepts for expressing possible algorithms in actual programs.

"Neural Computing"

John Hertz, NORDITA.

Neural computing is a new concept in computing. It is a concept biologically motivated and massively parallel. It has implications for both hardware and software. It will have application in the cognitive area including associative memory, recognition, error correction, and decision making.

A few key figures in the origin of neural computing are:

- McCullough and Pitts, 1943, for a network of binary threshold units
- R. Rosenblatt, 1960, for learning in perceptions
- E. Cainiello, 1961
- B. Widrow, 1962.

The things that make possible further progress today are:

- Progress in very-large-scale integration
- Progress in neuroscience
- Progress in behavior of large complex systems of interconnecting units.

In biological systems cells receive electrical pulses from other cells and each pulse raises the potential inside the cell, depending on the strength of the synaptic connection; when the potential in a cell becomes greater than a threshold value the cell fires a pulse of fixed strength along the axon. This results in a raising or lowering of the potential in a node of cells.

Formally, a neuron is a two-state system, either firing or not firing. A neural computing system differs fundamentally from a conventional computer:

- *It is more massively parallel*
- It is essentially collective – no programing of individual cells
- The program is contained in the synaptic connections
- It is robust against noise and errors
- A few errors can be tolerated.

The neural computer's main application will be in cognitive computation. The structure of the network is highly important. The formulation allows a new kind of biological modeling.

In physics terms, the dynamics of a spin system with energy,

$$E = -1/2 \sum_{ij} J_{ij} S_i S_j - \sum_i h_i S_i$$

i.e., field $h_i + \sum_j J_{ij} S_j$ acting on S_i

Moving toward states of lower energy.

A random mixture of plus and minus synapses is equivalent to spin glass, except $J_{ij} \neq J_{ji}$

Many metastable configurations, firing patterns, produce synaptic noise. To get from one pattern to another a soft threshold is introduced – i.e., probability of firing is increased.

Contributed Papers

As stated in my introduction to this report, the contributed papers and student scholarship papers will be summarized herein using the speakers' own abstracts.

"Supporting Distributed Matrix Operations on a Hypercube"

Clifford Addison et al., Chr. Michelsen Institute, Bergen, Norway.

The CMI High Level Library is a package of routines for a message-passing multiprocessor. It was originally designed to relieve the programer of the details of communication and data handling, especially in numerical finite difference computations. We are extending the library to support a general set of distributed data structures and operations for matrix and vector computations.

The object is to offer the programer a distributed implementation of the abstract data types – "matrix" and "vector" – that is as easy to use as the conventional single-processor implementation for matrices and vectors in terms of arrays. Actually the library will supply a choice of distributed representation – by row, by column, by block, dense or sparse – but the programer can choose the desired representation on grounds of efficiency, and then ignore the details of its implementation (or even change the choice later if necessary).

"Algorithms is for a Specialized Matrix Systolic Processor"

L.G. Aleksandrov et al., Center for Informatics and Computer Technology, Bulgaria.

The paper concerns some algorithmical aspects of a project for developing a high-performance specialized processor for fast matrix operations using a systolic array. The architecture of the processor is briefly described and implementations of different linear algebra algorithms exploiting the potential parallelism of the system are considered. The algorithms include solving systems of linear equations by the Jordan and the Gauss methods, LU and QR decompositions, matrix multiplication, and others. Despite of the simplicity of the architecture and the low technology requirements, the implementations have the following advantages:

- The processor solves problems with arbitrarily big sizes (limited only by the amount of memory).
- The utilization of the cells of the systolic array is very close to one.
- Numerically stable versions of the algorithms can be implemented.
- The range of solvable problems is large enough to cover important application areas.

"The CESAR Processor"

Vidar S. Anderson, Norwegian Defense Research Establishment, Norway.

CESAR is a parallel processor programable on various levels. It may be attached to any 32-bit host computer from Norsk Data through the standard DOMINO DMA Controller and the MultiFunction Bus Memory. CESAR has a peak performance of 320 MFLOPS and is relatively compact, implemented on 13 printed circuit boards occupying about half a card crate. As an example, 1K complex FFT's are computed in 0.257 milliseconds on average.

CESAR is well suited for tasks requiring intensive computations not dependent on the data content. Some signal processing algorithms are typical examples.

The talk is intended to give an introduction to the CESAR processor. Hardware modules as well as software tools are described from an application programer's point of view. This knowledge is necessary to understand the related poster "Signal processing with CESAR" by E-A Herland.

"A Benchmark Code for Multiprocessor Vector Supercomputers"

David V. Andersen, National Magnetic Fusion Energy Computer Center, California, and Ralf Gruber and Alexandre Roy, Centre de Recherche en Physique des Plasmas, Switzerland.

In the comparison of supercomputer performance one preferably seeks criteria that are relevant to the in-

tended applications. For example, large classes of problems from physics and other disciplines often result in very large systems of linear equations. In this regard, a program which solves such a system efficiently can be used as a benchmark to make comparisons among available and prototypical machines. We have developed the program PAMS (Parallelized Matrix Solver) which uses vectorization and multitasking (simultaneously) to solve a problem that arose in a 3-D plasma physics application. For this problem the matrix structure is tridiagonal block-banded with dense blocks.

The code employs a cyclic reduction procedure on the blocks which allows one to obtain an algorithm that is potentially very fast on multiprocessor vector supercomputers. The block-banded system (with dense blocks) is also encountered in other applications as well and therefore can be regarded as a good reference problem. Results from testing PAMS on the CRAY X-MP, CRAY-2, NEC SX-2, Fujitsu VP-200, and the CDC-205 will be presented. We also intend to present results from ETA-10 tests if we can gain access to the prototype machine. The value of PAMS as a benchmark for future more massively parallel computers will be discussed.

"Divide-and-Conquer Algorithms for the Computation of the SVD of Bidiagonal Matrices"

Dr. Peter Arbenz, Institut für Informatik, Switzerland.

Recently the divide-and-conquer algorithm proposed by Cuppen for the computation of the spectral decomposition of symmetric tridiagonal matrices has gained considerable interest due to the revision and successful implementation of Dongarra and Sorensen. Since the singular value decomposition of a bidiagonal matrix is closely related to the spectral decomposition of the tridiagonal $B^t B$ or BB^t but also of

$$OB$$

$$B^tO$$

there are several possibilities on how to apply the divide-and-conquer algorithm on the singular value decomposition. In this talk we present and compare numerically some old and new approaches.

"Lattice QCD-As a Large Scale Scientific Computation"

Clive F. Baillie, et al., California Institute of Technology Concurrent Computation Project, Pasadena.

Lattice QCD (Quantum Chromo-dynamics) is one of the most computationally intensive large-scale scientific computations. It can therefore be made to run efficiently on any computer. As part of the Concurrent Supercomputing Initiative at Caltech (CSIC), we have benchmarked Lattice QCD on a large number of computers: CrayX-MP and Cray 2 (vector supercomputers); Caltech/JPL Mark III, Intel iPSC, and Ncube hypercubes (MIMD shared distributed memory computers); and BBN Butterfly, Sequent Balance, and Alliant FX/8 (MIMD shared memory computers); and TMC Connection Machine 2, and AMT Distributed Array Processor (SIMD computers). Herein we explain the computation required for Lattice QCD, describe and contrast the different concurrent supercomputers used, and present the results of the Lattice QCD benchmarks.

"On the Performance of Shared Cache for Multiprocessor Organizations"

G.M. Chaudhry and J.S. Bedi, Wayne State University, Indiana.

High-speed computers use cache memories to increase the instruction execution rate by holding temporarily those portions of the main memory which are currently in use. A memory reference is a hit or a miss if the referenced datum is present or absent in the cache, respectively. After a miss the block containing the desired datum is copied from the main memory to cache memory. The hit ratio is the fraction of hits among all references; the miss ratio is the fraction of misses. In order to function effectively, cache memories must be carefully designed and implemented.

This paper studies the effects of shared cache on the performance of tightly-coupled multiprocessor systems in which main memory is also shared by all the processors. In shared cache, each processor is able to access a single cache, shared among all processors. The private cache and multicache systems suffer from data coherence problem. Another problem of the private cache is that certain shared system resources, such as operating system routines, may be copied several times in the cache memories when they are referenced by more than one process. Shared cache allows dynamic allocation of total cache space among the processors as compared to fixed cache allocation per processor in private cache systems.

"The Vectorization and Parallelization of ABAQUS".

R. Bell, IBM, UK, and B. Karlsson, Hibbert, Karlsson and Sorrenson, Inc., Providence, Rhode Island.

ABAQUS is a finite element structural analysis package marketed by Hibbert, Karlsson and Sorrenson Inc., of Providence, Rhode Island. One of its strengths is the analysis of nonlinear problems. ABAQUS is now available in a version that has been extensively vectorized for the IBM 3090 Vector Facility. In addition, elapsed times have been reduced by using 3090 central and expanded storage to keep the data arrays in storage rather than using DASD files.

CPU speedups relative to scalar in excess of 3.0 have been achieved together with elapsed time reductions in excess of 6.0. In addition, a two-way parallel version has been successfully demonstrated but not yet made commercially available. Many parts of the code have been vectorized but the most significant CPU speedups have come from the wavefront solver routine. This was extensively restructured so as to cast the FORTRAN in a form that would make maximum use of the IBM VF compound operation multiply/add.

This paper describes the techniques used in the vectorization work as well as the parallelization and bringing of files in storage. Performance results are also presented. The work was done by Hibbert, Karlsson and Sorrenson, Inc. with the cooperation of IBM, as part of IBM's worldwide program to assist vendors to enable their packages for the IBM 3090 Vector Facility. Technical guidance on the IBM Vector Facility hardware and software was provided by IBM UK Ltd.'s Technical Support function and also by the IBM Dallas Center.

"Use of Processor Networks for Parallel Polynomial Root Computing"

Ph. Berger and F. Hoxha, Department Informatique ENSEEIHT, France.

Solving polynomial equations is one of the oldest problem in algebra. A large number of algorithms, to compute the zeros of polynomials, have been developed. However, the problem of improving them remains current because the user's requirements (in the field of signal processing, C.A.D...) become more specific. A high precision is generally required. Furthermore, a minimal computation time is to be wished for (real-computation time constraint).

The use of parallel computers may be a reply of the last request. On the one hand, in many problems currently treated, the data size is not very large, and the implementation to supercomputers, whose peak performance are some hundreds of Megaflops, seemed unnecessary. On the other hand, for many users it is very difficult to have access to such computers. In this context, the development of methods on a processor network, like hypercube for example, may be interesting (good price/peak performance rate).

The object of this paper is to present some experiments in concurrently computing the roots of a high-degree polynomial on a network of transputers. The numerical algorithm allows determination simultaneously of all the roots of a one-variable polynomial, whose coefficients are real or complex numbers. A specificity of the method is the fact that a process can be associated with the computation of one or more roots, and then the program is easily distributed on a set of weak-coupled processors. The principal choices of implementation are linked with the strategy of data communication, the complexity of synchronization, and the type of network. For this aim, several versions are generated and compared.

Accordingly, we shall present the resolution algorithm in its mathematical context; afterwards, the criterions which have led us to elaborate different versions; and finally, numerical results and performances related to physical problems, problems drawn from literature, and randomly generated problems (polynomials of degree ≤ 64). We shall conclude on the efficiency of such a configuration in the field of numerical resolution of polynomial equations.

"The Spectrum of Sums of Projections with Application to Parallel Algorithms in Grid Refinement and Domain Decomposition"

Petter Bjørstad and Jan Mandel, University of Bergen, Norway.

Knowledge of the spectrum of sums of orthogonal projections can be used to estimate the rate of convergence of iterative methods based on additive formulations of the underlying problem. These methods are interesting for use in a parallel processing environment.

We give a precise characterization in the case of two projects, and show how the theory applies to both domain decomposition methods and to algorithms for grid refinement.

Numerical results form an Alliant FX/8 system will be presented.

"Block Cholesky Factorization of Large Sparse Matrices Parallel Computers"

Jon Braekhus, Veritas SESAM Systems, Norway.

A parallel block Cholesky solver is developed based on the secondary storage block Cholesky solver of SESAM. The new solver is also a secondary storage solver, but is prepared to take advantage of large memory. The solver consists of Cholesky decomposition and back substitution.

The solver performs first a symbolic factorization on block level to determine what block operation will take place. Then the dependencies between the block operations are found. This enables easy changing of the sequence to distribute the tasks and thereby of studying the load balance.

This new solver is tested on shared memory multiprocessors: Cray-XMP, Alliant FX, and VAX. Differences due to hardware and operating systems are studied.

The experiments on this solver includes variation of several parameters, the most important being:

- Block size (hereby setting typical vector length, work per processor and file storage requirements)
- Sequence of block operations done in parallel (this will affect memory usage, I/O and load balancing)
- The available amount of primary memory compared to bandwidth

The results for block approach is compared to the results obtained for non-blocked system by Alan George et al.

The tests include several of the matrices from the Harwell Boeing "Sparse Matrix Test Collection" and two real SESAM analyses. The performance is compared to the sequential Cholesky solver of SESAM and a general sequential sparse matrix solver.

SESAM is also prepared for parallel execution on network connected computers on a higher level. This is presented by Anders Hvidsten.

"Linear Programing on a Local Memory Multiprocessor"

Richard Chamberlain, Intel Scientific Computers. Wilts, UK.

Minimizing a linear function subject to linear equalities and inequalities can be a time consuming problem when the number of variable is large. The standard method to solve these problems is the simplex method.

This paper investigates the use of a local memory multiprocessor to solve linear programing problems. The distribution of the data, the communications requirement, the need for duplicated work and the potential of using vector processors at each node are discussed. Numerical results on the Intel iPSC and the vector extended iPSC-VX are presented.

"Parallel Processing Techniques of the Euler Equations on the IBM 3090 VF Computer"

S.M. Chang, IBM Corporation, et al., US.

The purpose of this presentation is to discuss parallel processing techniques on fluid applications with the use of the IBM 3090 computer with Vector Facility. Through the introduction of a Clebsch transformation of the velocity field, an equivalent set of the Euler equations is obtained for solving steady, three-demensional, transonic flows. The resulting equations are solved by the finite element method employing a block-structure relaxation scheme.

The solution domain is subdivided into blocks, and the equations are solved in an uncoupled form for each block with appropriate Dirichlet and Neumann-type boundary conditions. In this study, IBM's Multitasking Facility (MTF) is applied to distribute block processing across multiple processors. MTF is a VS FORTRAN facility which provides the capability to exploit the IBM's MVS/Extended Architecture (MVS/XA) operating system to allow a single program to use more than one processor simultaneously.

The results of two cases using the IBM 3090-200 will be presented on the solution of Euler equations for the wing-body problem. The first case consists of 24 blocks with 8,700 nodes and the second problem consists of 40 blocks with 120,000 nodes. Solutions will be reviewed in terms of computational speed and the level of parallelism achieved within the computations.

"Graphical Interface for Large-Scale Numerical Computation"

Jeremy Cook, Chr. Michelsen Institute, Bergen, Norway.

It is well known that large-scale numerical computation generates indigestible amounts of output data. Development of such applications is therefore tedious and time consuming. As an aid to rapid development of this type of numerical application, a system is now available to help the programer build a user interface for his application in a comparatively short time.

A reasonable interface would consist of one or more graphics windows which display the output data from the back end processor. With a multiprocessor machine it should be possible to display output from individual processors or from the whole machine. There should also be a control panel where the user is able to enter file names for input and output data as well as controlling the flow of data and selecting which results to display by easy-to-use menu options.

It would typically require 5-25 mandays, depending on the level of complexity, to develop such a user interface for a parallel application with a graphics workstation at the front end. The goal for the system described is to enable the user to build a basic interface within a windowing environment in the course of the working day. With such an interactive graphics interface the numerical application will be developed much faster than normally attainable. The increase in productivity is difficult to measure but is estimated to be of order 5, or greater.

The pilot system, initially aimed at users with SUN-workstation/Hypercube combinations, is implemented for X-windows and the SunView windowing environment. The user interface is built by specifying the layout and function of each window in a configuration file. It is also necessary for the user to write a simple graphics function which will display the data in the best form for interpretation.

This work is part of a CMI project to build a user friendly environment for large-scale numerical computation.

"Fully Vectorizable Preconditionings for Parallel Local Grid Refinement"

J.C. Diaz et al., The University of Tulsa, Oklahoma.

Many time-dependent problems involve both general phenomena as well as significantly localized phenomena. These are often critical to the overall behavior of the physical processes and are usually dynamic in nature. For large-scale physical modeling, it is frequently impossible to use a uniform grid, in the numerical procedure, which is sufficiently fine to resolve the local phenomena without yielding an extremely large number of unknowns.

Use of dynamic grid refinement has been shown to be a practical method to approach these large problems. A coarse grid is placed over the domain and finer grids are used in those subregions where localized phenomena appear. In general, several levels of refinement might be necessary to achieve a given minimization of the error in the solution. Tree-like data structures permitting the efficient control of the placement and/or removal of the fine grids have been discussed by several authors.

In particular, a method permitting the placement or removal of overlapping grids has been shown to be effective when dealing with systems of hyperbolic conservation laws. Because of the nature of the data structure, grids at the same level in the tree are completely independent from each other and can be solved in parallel. This inherent coarse grain parallelism makes this method

even more attractive for utilization in the modeling of large-scale problems.

Our main interest is in transport-dominated diffusion problems which, in general, require the use of implicit-in-time discretization schemes. As a consequence, large, sparse, nonsymmetric systems of linear equations have to be solved in order to advance the solution for each independent grid.

To further exploit the capabilities of today's parallel vector-computers, it is imperative to have a scheme for solving the sparse nonsymmetric linear systems which can be fully vectorized. In this way, parallelism is achieved through the distribution of the grids among the parallel processes available, and also by making use of the vector operations for each parallel process. We use a conjugate-gradient-type method with preconditioning to solve the sparse nonsymmetric linear system arising from the discretization of the physical model.

The main obstacles for complete vectorization have been the preconditioning calculation, and the application step within the iteration. For the matrices obtained using the above point discretization operators, the existing preconditioners usually require a block-recursive procedure which prevents vectorization.

Preconditionings based on nested-incomplete-factorization and approximate inverses have been proposed by some authors. At the innermost level of incomplete-factorization the approximate inverse of a tridiagonal matrix is calculated. Preconditioning schemes for nonsymmtric problems using the Frobenius norm minimization for the determination of the approximate inverse, are discussed herein. We derive a formulation of this preconditioner which can be fully vectorized.

The application of this preconditioner, requires only matrix-vector and vector-vector products. It can be vectorized in full if appropriate data structures are used to present the sparse matrices.

Numerical experiments indicate that, for a class of nonsymmetric problems, application is up to 504 faster than existing methods, such as *ILU*.

Calculation of the preconditioning is somewhat more expensive, but the faster application and the reduced number of iterations necessary to minimize the error more than compensate for this drawback. Some new theoretical results concerning the properties of the approximate tridiagonal inverse have been obtained and will be presented.

Numerical results for some sample transport-dominated diffusion problems to illustrate the performance of the overall parallel method using vector-parallel architectures, will also be presented.

"Finite Element Optimisation in ADA Using Automatic Differentiation"

I.C.W. Dixon and M. Mohseninia, The Hatfield Polytechnic, UK.

In an earlier paper Dixon and Mohseninia (1987), the auth. rs described an implementation of Rall's (1981) automatic differentiation approach in ADA using the concepts of new data types and overwritten operators. In that paper, the approach was combined with the Truncated Newton optimization algorithm (Dembo and Steigaug, 1985), and tested on a number of simple test problems.

The automatic differentiation approach has now been extended to generate sparse Hessian matrices. The finite element optimization approach to nonlinear partial differential equations has been used to generate nonlinear optimization problems with large dimension, and results of applying the algorithm to such problems will be presented.

"Using Symmetries and Antisymmetries to Analyze a Parallel Multigrid Algorithm: The Elliptic Boundary Value Problem Case"

Craig C. Douglas and Barry F. Smith, US.

We exploit symmetry and antisymmetry properties of a class of elliptic partial differential equations to prove when a particular parallel multilevel algorithm is a direct method rather than the usual iterative method. No smoothing is required for this result. Examples are presented, including variable coefficient ones. A connection between our algorithm and domain decomposition is established, even though this algorithm is more general and different. We also analyze the parallel algorithm when it is iterative. We show how to increase processor utilization in this case. We analyze Hackbusch's so-called "robust multigrid" algorithm for some model problems and show that our parallel algorithm uses much less computer time with, at most, the same amount of storage.

"Parallel Implementation of the Boundary Element Method"

J.B. Drake et al., Oak Ridge Laboratory, Tennessee.

In this paper the implementation of the boundary element method (BEM) on a hypercube is considered. A program solving the three-dimensional Laplace equation for the electric field of an electroplating cell is described. The BEM is specifically adapted to this application, which requires the solution of nonlinear boundary conditions. As a consequence, the matrices associated with the prescribed Dirichlet values and the prescribed Neumann values are formed separately and stored. A step of the algorithm is thus required to combine and rearrange the system of equations into the standard form $AX=B$.

The matrix A is dense and nonsymmetric, and recent advances in the art of solving dense linear systems on hypercubes are taken into account in the development of the algorithm. Estimates of the arithmetic complexity at each step of the algorithm and model for the communication costs are used to study the parallel performance of the BEM is particularly well suited for parallel solution and can be implemented efficiently on a hypercube.

"Hypercube Implementation of a Linear Systems Solver Using Tensor Equivalents"

Lisette de Pillis et al., Chr. Michelsen Institute, Norway.

A new stationary iterative method for the solution to special linear systems developed by Dr. John de Pillis is implemented on an Intel iPSC-VX Hypercube. The method finds the solution vector x for the invertible n x n linear system $Ax = (I - B)x = f$ where A has real spectrum.

The solution method converges quickly because, through the use of tensor products, an equivalent system with a better spectrum is generated. The Jacobi iteration matrix b is replaced by the equivalent iteration matrix with a smaller spectral radius. A good approximation to the spectral boundaries of a is a requirement for this algorithm. A method for finding these spectral parameters in parallel is discussed. The parallel algorithm for finding the vector x partitions A row-wise among all the processors in order to keep memory load to a minimum and to avoid duplicate computations. The algorithm has been fine-tuned in order to take full advantage of the vector hardware on the hypercube and to further reduce run-time. Example problems and timings will be presented.

"Prospectus for the Development of a Linear Algebra Library for High-Performance Computers"

James Demmel et al., Courant Institute, New York.

We propose to design and implement a transportable linear algebra library in FORTRAN 77 for efficient use on a wide range of high-performance computers. The production of such a library for the most commonly encountered problems of linear algebra would have several benefits:

1. It would facilitate the development of scientific codes on high-performance computers. This area was recently identified by the Computational Science and Engineering Initiative of the National Science Foundation as in serious need of development. The large and growing variety of machine architecture puts a heavy burden on the scientific programer to use each machine efficiently, since speed is the major reason to use high-performance computers. The availability of a highly efficient library for standard linear algebra problems on each major machine would free the programer to work on more interesting parts of the code.

2. It would increase the portability of scientific codes between different computing environments. Programs written largely in terms of calls to a standard library would require less work to tune to the new computer architecture, since the library routines would already be tuned.

3. It would improve the utilization of a scarce resource. By making efficient, state-of-the-art codes available even to beginning users, more efficient use could be made of expensive supercomputer cycles.

4. It would provide tools to aid performance evaluation of computers. A national study has identified the evaluation of supercomputer performance as an area in need of development and standardization.

To realize these benefits, the new library must satisfy several criteria. First, the library must be highly efficient, or at least "tunable" to high efficiency, on each machine. Otherwise it will not be useful for benchmarking nor will it improve utilization, and users will continue to write their own (not necessarily better) algorithms. Second, the user interface must be uniform across machines. Otherwise much of the convenience of portability would be lost. Third, the programs must be widely available.

The success of the NETLIB facility has demonstrated how useful and important it is for these codes to be available easily, and preferably on line. We propose to distribute the new library in a similar way, for no cost or a nominal cost only. In addition, the programs must be well documented, in the style of the LINPACK manual. To achieve these goals, we propose a linear algebra library, based on the successful EISPACK and LINPACK libraries, with the following further developments:

- Integration of the two sets of algorithms into a unified library, with a systematic design
- Incorporation of recent algorithmic improvements
- Restructuring of the algorithms to make as much use as possible of the Basic Linear Algebra subprograms (BLAS). Use of the BLAS is the basis of our approach to achieving efficiency, and is discussed at greater length in section 2.2.

In short, a library would become a central part of the infrastructure of a growing high-performance scientific programing environment, much as conventional libraries for serial machines are essential to conventional scientific computing.

"Functional Languages for Scientific Software"

Lennart Edblom, Institute of Information Processing, University of Umea, Sweden.

During the past few years a number of different parallel computers have appeared. They all try to exploit parallelism one way or another. The architectures of these computers are however widely differing, and there is no common language or language features for programing parallel computers.

We examine some of the problems associated with programing current parallel computer architectures. Regardless of whether you are using a multiprocessor with distributed memory or a vector computer, you must be aware of the overall organization, and also many of the particular details of the computer system to use it efficiently. We also examine some aspects of the languages currently used for programing parallel computers, and find that they are quite inadequate to express parallelism in a machine-independent but problem-oriented style.

Our conclusion is that these problems are best solved by introducing radically new architectures and languages. We propose data flow architectures and functional languages as a possible solution. One property of functional

languages is that there is no inherent sequentiality in the language. Concurrency is implicit in a functional program, and both regular and irregular parallelism, both operator-level and process-level parallelism are equally well represented. Similarly, data flow architectures have no concept of control flow. An instruction is ready for execution when its operand has arrived, thus highly concurrent computation is possible.

We have chosen scientific software as our application area, in particular linear algebra. We show how some well-known linear algebra problems may be coded in a functional language. The examples include matrix multiplication, gaussian elimination and Cholesky factorization. Functional languages are found to have several advantages, e.g., that the functions are formulated on a high level and are amenable to program transformation. Furthermore, a compiler can easily extract a suitable gain of parallelism.

We will continue to investigate how a functional language for the development and coding of scientific software should be designed. There are several remaining problems, e.g. the efficient handling of arrays, but the potential advantages are more than enough to motivate continued research.

"Coherent Parallel C"

Edward W. Felten et al., California Institute of Technology, Pasadena.

Coherent Parallel C (CPC) is an extension of C for parallelism. The extensions are not simply parallel for loops; instead, a data parallel programing model is adopted. This means that one has an entire process for each data object. An example of an "object" is one mesh point in a finite element solver. How the processes are actually distributed on a parallel machine is transparent – the user is to imagine that an entire processor in a distributed-memory environment is dedicated to each process. This simplifies programing tremendously: complex if statements associated with domain boundaries disappear; problems which do not exactly match the machine size and irregular boundaries are all handled transparently.

The usual communication calls are not seen at all at the user level. Variables of other processes (which may or may not be on another processor) are merely accessed (global memory). The first pass of the CPC compiler schedules the necessary communications in an efficient, loosely synchronous manner. Processes in CPC are insulated from one another and interact in a deterministic manner. This allows tractable debugging. Standard C I/O is provided, with simple extensions for parallelism.

Naturally, some performance must be sacrificed for programing ease. Linear and near-linear speedups still occur, although with a lower level of absolute performance. Results and performance models will be given. CPC is not specific to distributed memory machines. At the user level, one sees only processes and knows nothing of domain boundaries, processor numbers, etc. Implementation of this language on other architectures is natural – there seem to be no fundamental problems with CPC on shared-memory parallel computers or fine-grained SIMD computers.

"Chess on a Hypercube"

Edward W. Felten et al., California Institute of Technology, Pasadena.

We have implemented computer chess on an NCUBE Hypercube. The program follows the strategy of currently successful sequential chess programs: searching of an alpha-beta pruned game tree, iterative deepening, transposition and history tables, specialized endgame evaluators, and so on. The search tree is decomposed into the hypercube using a recursive version of the principal-variation-splitting algorithm. Roughly speaking, subtrees are searched by teams of processors in a self-scheduled manner. Search times for related subtrees vary widely (up to a factor of 100), so dynamic reconfiguration of processors is necessary to concentrate on "hot spots" in the tree.

An interesting feature is the global transposition table. For this data structure the hypercube is used as a shared-memory machine. Multiple writes to the same location are resolved using a priority system which decides which entry is of more value to the program. Implementation of the transposition table as "smart" shared memory is crucial to the performance of the program.

The program has played in several tournaments, facing both computers and people. Most recently it scored 2-2 in the North American Computer Chess Championship.

"Local Convergence of Nonlinear Multisplitting Methods"

Dr. Frommer, Universität Karlsruhe, West Germany.

Multisplitting methods for the solution of a system of linear equations $Ax = b$ are based on several splittings of the coefficient matrix A. In a parallel-computing environment each processor performs iterations corresponding to one of the splittings, and the final iterate is obtained by combining the individual iterates in an appropriate manner.

In a systematic way we now extend the idea of solving a nonlinear system of equations $F(x)O$. These nonlinear multisplittings are based on several nonlinear splittings of the function F and the corresponding calculations can again be performed in parallel. Each processor would now have to calculate the exact solution of an individual nonlinear system belonging to "his" nonlinear multisplitting. Although these individual systems are usually much less involved than the original system, the exact solutions will in general not be available.

Therefore, we consider important variants where the exact solutions of the individual systems are approxi-

mated by some standard method such as Newton's method.

We present a local convergence analysis of the nonlinear multisplitting methods and their variants. In particular we will show that these methods converge linearly and that the speed of convergence is determined by an induced linear multisplitting of the Jacobian of F. It will also turn out that the speed of convergence is not affected if the individual systems are solved exactly or only approximately via Newton's method. We include some numerical experiments to illustrate our results.

"A Parallel Computer Implementation in Finite Element Methods"

Robert E. Fulton et al., Georgia Institute of Technology, Atlanta.

The paper reports on the development and implementation of parallel processing software for finite element solutions. A parallel FEM equation solver has been developed and tested for several static and dynamic analysis demonstration problems. It has also been incorporated in the production finite element system FENRIS, with applications to crash dynamic test problems. Parallel processing methods for transient analysis have been studied using implicit and explicit numerical integration schemes. Parallel software has been implemented on several multiprocessor machines, including shared memory and local memory computer architectures. The results show that a parallel processing approach can significantly reduce execution time for large-scale finite element problems.

"Numerical Sea Modelling Using Parallel Vector Processing"

G. Furnes et al., Bergen Scientific Centre, IBM, and Institute of Marine Research, Norway.

Three-dimensional hydrodynamic equations for tides and wind-induced flow in a sea region are solved numerically using two different computational techniques; first by using a single-processor computer and then on a parallel computer with a number of processors.

The model equations are solved explicitly on a finite difference staggered grid in the horizontal space domain. In the vertical domain both expansion in terms of eigenfuctions and finite difference box schemes are considered. In the time domain we used forward time stepping. The parallel processing scheme described in this paper consists roughly of dividing the sea area into a number of subdomains determined by the number of processors available.

Experiments with different horizontal resolutions in the "functional model" and the "grid box model" are performed, and the relative parallel efficiency will be discussed.

"The Evolution of Parallel Processing at CRAY Research"

Mark Furtney, CRAY Research, Mendota Heights, Minnesota.

In 1983, CRAY Research introduced, the X-MP/2, and the face of supercomputing has never been the same since. Since that time, 4-CPU and 8-CPU machines have been introduced. This talk briefly describes the hardware organization which promotes these first commercially successful multiprocessor supercomputers but concentrates on the evolution of the support software which has grown to deliver hardware performance to users. The first effort (now termed Macrotasking) provided a library of FORTRAN-callable routines which implemented a set of synchronization primitives with which users could create and control multiple tasks within a single program. This library soon became a defacto industry standard, but it did not fulfill all the need of the supercomputing user community.

Microtasking evolved from Macrotasking, and its very low overhead synchronization allowed new levels of parallelism to be profitably exploited. The design of Microtasking will be covered in some detail, including a discussion of why it works so well for both batch- and dedicated-mode computing, and why it has gotten so popular. The next step in the evolution of parallel processing software (termed Autotasking) will then be described again including a discussion of software design issues, considerations, tradeoffs, and decisions.

"The MMX Parallel Operating System and its Processor"

Eran Gabber, Tel Aviv University, Israel.

MMX (Multiprocessor Multitasking eXecutive) is a small yet powerful operating system for shared memory multiprocessors. The MMX parallel processor is a small shared bus multiprocessor assembled from several National Semiconductor processor boards. Together, MMX and its parallel processor provide a flexible and powerful testbed for parallel software development.

This paper describes MMX structure, services and performance. Parallel programing methods using MMX are sketched along with timing and speedup measurements of several parallel programs. The paper concludes with a brief description of future research directions.

"The Arithmetic Mean Method for Solving Linear Dissipative Systems on a Vector Computer"

Ilio Galligani and Valeria Ruggiero, Universities of Bologna and Ferrara, Italy.

This paper is concerned with the implementation on a parallel computer with a few vector processors of the *arithmetic mean method* for solving dissipative system of the form

$$\frac{du(t)}{dt} + A.V(t) = b \quad t > O$$

$$V(0) = g$$

where the matrix $A+A^T$ is symmetric positive definite. For example, such systems arise in solving the initial-boundary value problem for the diffusion-convection equation on a rectangular domain by the method of lines.

In this case, A is a large and sparse with a nonrandom sparsity pattern. In this note we make the assumption that the matrix A can be expressed as $A=A_1+A_2$, where A_k (or PA_kP^T, with P a permutation matrix) is a matrix of simple structure (for example, triangular or tridiagonal). Then, it is possible to solve the system with the arithmetic mean method. The consistency and the stability of this method have been analysed.

The method is well suitable for parallel implementation on a multiprocessor system that can execute concurrently different tasks on a few vector processors with shared central memory, such as the CRAY X-MP/48. A high-level parallelism among independent tasks is offered by the Cray multitasking. An implementation on CRAY X-MP/48, using microtasking directives, of the method has been developed when A is a block-tridiagonal matrix and each square block submatrix on the diagonal of A is a tridiagonal matrix. A detailed description of this implementation is given and the results of some computational experiments carried out on test block-tridiagonal matrices are reported.

"Parallelizing an Efficient Partial Pivoting Algorithm"

John R. Gilbert, Cornell University, New York.

A sparse matrix can be factored by Gaussian elimination with partial pivoting in time proportional to the number of nonzero arithmetic operations, using an algorithm of Gilbert and Peierls. A sequential implementation of that algorithm is quite efficient in practice.

We obtain a shared-memory parallel version of the algorithm by using two ideas: Elimination trees are used to identify parts of the factorization that can be performed independently in parallel, and the graph-theoretic structure prediction step in the original algorithm is modified to allow pipelining of consecutive columns. We present results from an experimental implementation on an Alliant FX/8 multiprocessor.

"Parallel Neural Network Simulation Using Sparse Matrix Techniques"

Jeremy Cook et al., Chr. Michelsen Institute, Norway.

Neural computing is an emerging concept in artificial intelligence. This new way of programing attempts to simulate the way in which the brain processes information. The massive parallelism of the brain makes human perception much faster than pattern recognition algorithms on conventional computers. Neural networks are a natural framework in which to implement applications such as data bases, character recognition, speech recognition, and syntax checking.

Real neural computers capable of significant computation do not exist yet, but today's conventional parallel comp ters are a good testbed on which to simulate neural computers and experiment with neural algorithms. This paper reports on the use of a message-passing multiprocessor to simulate a neural computer.

Simulating a neural network efficiently on a multiprocessor is related to parallel sparse matrix computation. One basic iteration of the network is essentially a matrix-vector multiplication, with addition replaced by evaluation of a nonlinear threshold function that models the response of a neuron. The network is sparse; most pairs of neurons are not connected at any given time. Communication and load-balancing issues are similar to those in numerical sparse matrix computation. However, a learning network must also periodically modify its connection weights (that is, the matrix values), or even its connectivity (that is, the matrix structure).

We shall describe experiments with a neural network simulation on the Intel iPSC hypercube machine. The implementation is based on a combination of standard sparse matrix technology and dynamic restructuring of the network during the course of the computation.

"Image Analysis Algorithms on Supercomputers"

Fred Godtliebsen et al., The Norwegian Institute of Technology, Norway.

Iterated Conditional Modes and Simulated Annealing are two standard statistical techniques for image improvement in image analysis. They may, however, be very time consuming.

The algorithms are applied in medical diagnosis.

This paper gives implementation and examples tested on vector and parallel computers.

The algorithms are developed on a CRAY X-MP/28. We also plan to run them on VAX8600, Apollo dn580 and Alliant FX/6. Speedup-factors and execution times are given and discussed.

"Optimal Power Scheduling of a Large Electric Network Via Nonlinear Programing on the CRAY X-MP/48"

L. Grandinetti and D. Conforti, Universita della Calabria, Italy.

A problem of great practical interest, related to the production, transmission and distribution of electric energy in a network, is taken into consideration and the optimal management policy for the system is formulated as a nonlinear mathematical program. This program is characterized by large-scale dimension and highly nonlinear constraints; the need for a "real-time" numerical solution is an additional distinctive aspect of it.

A number of mathematically sound nonlinear optimization algorithms, which use gradient information of the objective and constraint functions, is selected with a special attention to those particularly suitable for a proper matching to the resources offered by a vector supercomputer.

Analysis of numerical results suggests that the computation performed on a CRAY X-MP/48, provides a satisfactorily efficient solution of the proposed problem, in spite of its severe computational characteristics.

"An Extension of NAG/SERC Finite Element Library for Message Passing Multi-Processor Systems"

C. Greenough and C.J. Hunt, Rutherford Appleton Laboratory, UK.

During a time when concurrent computing hardware is developing quickly and software costs are escalating it is important to develop programing methodologies that use a significant amount of existing serial software on these systems.

In this paper we present a number of extensions to the serial version of the NAG/SERC Finite Element Library which will enable the users of the serial Library to make use of the many emerging message passing concurrent systems.

Under the basic requirement that the use of existing serial user programs should be maximized and that the general philosophy of program should not change, a number of extension have been developed to aid users in exploiting multiprocessor systems.

The paper will address two areas: the programing philosophy and the implementation of the finite element method. A discussion of the method used for domain decomposition, element and system matrix assembly and linear algebra in relation to processor usage will be given.

These extensions have been designed for a general multi-processor system and have initially implemented on a hypercube architecture and some results using the extended system will be given. Some indication of future work will given particularly in the use of transputer systems.

"An Array Processor Architecture for Neural Networks Analysis"

Anne Guerin et al., Institute National Polytechnique de Grenoble, France.

In the last few years, neural networks analysis has developed astonishingly. According to this approach, the study of existing functions in the nervous system demands some powerful simulation tools. It is obvious that some processes usually are computed with effectiveness (perception for example) in the nervous system, the same are controlled with difficulty by actual processors.

The nervous system organization is different, is totally opposite the computer principles in von Neuman's classical processing architecture. At the very first level, the nervous system is composed of highly interconnected neural networks. So the two main characteristics are, on one hand, a great number of simple cells (neurons), on the other hand, a great degree of interconnection between these cells. So this structure requires power more for communication control than for computation in each cell. Taking inspiration from the nervous system organization, the so-called "neuromimetic" architectures provide an optimal combination of power and speed. With the improvement of VLSI technology, it is quite easy to implement operative cells, but the challenge is to control their full interconnection.

An alternative to these problems is to build a calculator according to the suitable architecture, which must provide a good compromise between a general-purpose and a dedicated computer in the class of parallel processors. That is to say, our aim is not to implement in VLSI a structure of neural networks directly, but it is to create an efficient arithmetic configuration for simulation of "neuromimetic" networks. We propose an array processor architecture with a very simple interconnection network between the processing slices (processing element with associated memories). In fact, this interconnection network must be elementary, because for neural network analysis, both scalar and vector processing abilities are required together. These algorithms compute upon well-structured data flows in relation with a big amount of data memorized in all the processing slices.

To be efficient, both for scalar and vector processing, the arithmetic structure must be reconfigurable. So we chose the simplest arithmetic array: a one-way linear array which is efficient for matrix and vector multiplications (basic operations in neural networks). For vector processing, the arithmetic configuration is a pipeline chain of processing slices. For scalar computations, we only break the linear array. So each processing element in a parallel and autonomous way computes the complete scalar equations in relation with its memory.

At last, in this article, we describe a calculator named "CRASY," which has reconfigurable architecture. The processor CRASY, as a prototype, is composed of only two slices, each is able to perform 20 Mflops. The performance of the calculator is directly proportional with the number of processing slices. This modular architecture is very useful for the extending the processor. It is easy to build a N-slice processor able to perform 20xN Mflops. We plan to use CRASY's computational power in the simulation of learning neural networks models which constitute a new and efficient way of adaptive information processing.

"Parallel Multigrid Solver for 3-D Anisotropic Elliptic Problems"

Ute Gartel, Center for Computer Science, West Germany.

The efficiency of multigrids in solving elliptic partial differential equations depends essentially on the ability of the "smoothing operator" to reduce high-frequency error components. Whereas for isotropic 3-D problems, pointwise relaxation has reasonable smoothing properties, for anisotropic cases line or even plane relaxation has to be used.

A parallel (MIMD, local memory) multigrid program for solving 3-D problems with arbitrary anisotropies will be presented. Parallel line relaxation is based on a reduction method, parallel plane relaxation is implemented by using suitable 2-D multigrid methods. Numerical results, especially concerning the performance will be represented and discussed.

Highest possible portability was one major goal in the program design. This is achieved by means of a general and flexible library of "communication routines" which do both the mapping and the communication. Machine-dependent language constructs are completely hidden inside the library routines. This way, the program can be used on different parallel and even sequential machines by simply adapting the communication routines.

"The Use of Systolic Arrays for Finite Element Calculations"

Dr. Linda J. Hayes, University of Texas at Austin.

Systolic arrays are a network of very simple processors which operate in parallel and are usually designed to be special-purpose systems. One characterization of systolic arrays is their asynchronous operation. The results are passed between processors as data tokens and each input travels through an array of cells before a final result is returned to memory.

A systolic array design will be presented for doing finite element calculations. In this array each processor is extremely simple and there is one processor allocated for each node in the finite grid. A single systolic array design will be used not only to generate the finite element equations but to maintain them at either an element or a global level and also to solve the resulting linear systems of equations. Each processor maintains one row of the coefficient matrix either in element or global form.

Connectivity and data flow between processors is dictated by the connectivity of nodes in the finite element grid. The Hypercube was used to simulate the systolic array design, and results are presented for several test cases.

"Vectorization of Arnoldi-Tchebychev Method for Nonsymmetric Matrices"

F. Chatelin et al., IBM Scientific Center, France.

The vectorization of the Arnoldi-Tchebychev procedure for solving nonsymmetric eigenvalue problems is discussed. The procedure is based on the iterative Arnoldi method in conjunction with the Tchebychev acceleration technique. New criteria are established to identify the optimal Tchebychev ellipse of the eigenspectrum.

A simple method has been developed to determine the parameters of the optimal ellipse passing through two eigenvalues in a complex plane relative to a reference of complex eigenvalue. The algorithm is fast, reliable, and does not require a search for all possible ellipses which enclose the spectrum. The procedure is applicable to nonsymmetric linear systems as well.

"Applications of Computational Fluid Dynamics for External Flows Relevant to Offshore Engineering Employing Supercomputers"

M. Bercovier et al., The Hebrew University, Israel.

The application of computational fluid dynamics has been shown to enhance the design process for offshore structures. Due to the geometrical complexity of such structures, the numerical models are frequently comprehensive three-dimensional models which entail the use of supercomputer capacity in order to ensure solutions within the tight project schedules required by the offshore industry.

The present paper deals with the numerical solutions to the incompressible Navier Stokes equations for external, wind-induced flows. These flows are of significance in terms of environmental, safety, and loading aspects for offshore structures.

The program used in this study solves the Navier Stokes and continuity equations using the finite element method (FEM). The FEM has certain advantages in the generation of the mesh for the complex geometries and boundary conditions necessary for the above applications.

The flow fields obtained using the methods are displayed together with a brief discussion on the relevant turbulence models. The development time for obtaining the results both in terms of manhours and computational effort is also discussed and compared with alternative experimental methods.

"Aspects of Sparse Matrix Technique on a Vector Computer"

Niels Houbak, Lab. for Energiteknik, Denmark.

Sparse Matrix Technique is most profitable when rows only contain few non-zeros (i.e., short vector length) whereas the advantages of the vectorcomputers are most evident for long vectors. This, as well as the normally not vectorizeable overhead in the sparse codes, gives rise to the impression that sparse codes vectorize poorly.

In many applications though – e.g., solving stiff systems of ODE's – it is often the case that many (almost) identical matrices have to be factorized in the same run. Exploiting the fact that the structure of the LU-factors only need to be computed once (or only a few time) one can dramatically reduce the overhead and increase the vectorizeability of all the factorizations but the first. Increasing the storage requirements, one may even reduce the overhead for the third and the following factorizations.

The various aspects hereof will be illustrated by runs made on a CRAY-XMP and on an Amdahl/VP1100.

"A Dynamic Load Balancing Scheme to Utilize the Parallelism in a 'FE' Structural Analysis Program"

Anders Hvidsten, University of Bergen, Norway.

The structural analysis program SESAM supporting a multilevel substructuring techniques is being parallelized. The parallel version of this program is designed to run on different computer architectures, including both a distributed net of computers and shared memory multiprocessors.

The approach taken is to view all user-defined substructures in a structural model as subtasks. The actual

subtasks to be performed are factorization and computation of Schur complements when traversing in the hierarchy towards the top, and back substitution when traversing downward in the hierarchy after equation solving. These subtasks are organized in a shared pool of work. Dynamic scheduling of subtasks in distributed and nonuniform sets of computers is investigated, and different objective functions are proposed for achieving good load balancing.

The pool of work is implemented in terms of a shared mailbox. This shared mailbox is the only way processes are allowed to communicate with each other. Functionality and performance of various communication utility packages used to implement interprocess mailboxes are analyzed.

The analysis is supported with examples from two parallel environments: (1) a distributed net including Sun3/'s and Alliant FX/8 connected via an Ethernet, and (2) a Cray XMP with two processors.

"Improvements to the Black-Oil Simulator (Eclipse 100)"

Oddvar Gjerde, IBM Oslo, Norway.

Parallelization of a black oil simulator based on isolated geologic structures in the linear solver has been proposed by Kaarstad et al.

Firstly, we have succeeded in making parallelized sections to take into account the possible number of phases present in each reservoir. In the previous implementation if one reservoir had three phases present, then all reservoirs were treated as three-phase. In the new implementation, we can utilize the fact that if any reservoir has only two phases present, it will be treated as such. This will reduce the number of equations to be solved for the two-phase case by factor of four-ninths.

The latest implementation has taken the above mentioned proposals a step further. In order to simplify this by the need to add minimum code, we have renumbered not only the cells in each reservoir, as was the case in the above mentioned report, but also the planes so that each reservoir has its own plane count, similar to the pointers for the cells. This means that by pointing to the first and last planes in each reservoir, much of the code remains as before except for the length of the vectors. Thus it is not necessary to change existing pointers from vectors to two-dimensional arrays.

Furthermore, we have extended this method to the routines which construct the linear equations using the same approach.

Lastly, we are also adding new parameters to define reservoir boundaries in a more general way, so that they do not need to consist of a single box delineated by vertical and horizontal planes.

"Parallel Implementation Techniques for Prolog on the DAP"

P. Kacsuk, Computer Research and Innovation Center, Hungary.

The main results of a research on the parallel implementation of Prolog on the distributed array processor (DAP) is described. Though many projects are under way to implement Prolong on MIMD computers, there have so far been no proposals for implementing Prolong on SIMD machines.

The underlying project proved four different approaches to be viable for implementing Prolog on the DAP.

1. Basically sequential implementation mode where only certain parts of the Prolog interpreter work in parallel. An example for the parallel subactivities is the undo mechanism during backtracking.

2. Applying a set-oriented interpretation mechanism, where a mixed depth-first/breath-first search strategy is adopted. In this strategy the multiple-fact branches of a conventional Prolog search-tree are considered as generating binding sets rather than search non-determinism.

3. SIMD machines such as the DAP are efficient at data-parallel rather than task-parallel problems, enabling them to work efficiently with large, homogeneous data structures. Arrays are the most obvious software realization of the SIMD aggregate of processing elements. Extending Prolong with arrays enables Prolog to be used efficiently in applications with a large number part.

4. A cellular-dataflow model for executing logic programs was successfully implemented on the DAP. The close relationship between cellular automatas and DAP made it possible to implement the model in a straightforward and elegant way.

The implementation of a Prolog variant for the DAP, called DAP Prolog, is based on the above techniques. DAP Prolog is an extension of ordinary Prolog with homogeneous data structures.

DAP Prolog = Prolog + Sets + Arrays

The paper summarizes the main features of DAP Prolog and gives a detailed description of the parallel implementation techniques mentioned above.

"Debugging Support for Parallel Programs"

David W. Krumme et al., Tufts University, Medford, Massachusetts.

The first question in debugging is "What is my program doing?" All that is needed for a large portion of the debugging process is to answer basic questions regarding the flow of execution and the values of variables. On a multiprocessor with hundreds or thousands of computationaal nodes, extracting and utilizing this information poses special problems. Indeed, the difficulties of debugging programs on large mutiprocessors are a major impediment to the exploitation of the computational power that these machines offer.

This paper describes some specific debugging support tools for application programs developed as a part of a general research effort in parallel program environments of Tufts University. These tools are implemented on a 64-processor NCUBE hypercube.

In contrast to interactive debuggers which allow the programer to probe for specific state information, we are concerned here with tools that provide a general overall picture of an execution, without relying on the programer to determine what to probe for. We perceive a gap between the basic initial condition of total ignorance on the part of the programer and the situation when the programer knows enough about what is happening in an execution to probe intelligently for particular values. We see great utility in tools that allow a rapid progression from the former condition to the latter.

There are three problems to solve in developing a debugging tool of this sort: deciding what basic facts should be conveyed to the programer; providing instrumentation to collect the relevant data; and creating a user interface to present it efficiently. Our instrumentation is embedded in a custom operating system called SIMPLEX that we have designed and implemented on our machine; it is capable of measuring any quantity of interest, and in response to polling from a host monitor process it sends out requested data. The user interface, called SEE-CUBE, solves the problem of presenting large quantities of information to the user through carefully designed color graphics displays.

We distinguish between execution data that can be extracted automatically for an arbitrary program and that which depends on advance planning by the programer. Experience has confirmed our belief that a tool that does not require special action by the programer has important advantages over one that does. For example, in one case a programer noticed in a general display involving a supposedly fully debugged program that a troublesome interaction between nodes was occurring that was never suspected of being possible, and hence that would never have been probed for. When coordination among processors is organized around message transmissions or other coarse-grained events with operating system involvement, these events provide the basis for such automatically selected execution data.

Message transmissions are easily measured and they are significant to the programer since the program has been necessarily organized around them.

Our general status display uses the following automatically collected items:

- The computational state of each processor which is color coded to represent one of: computing, waiting to read a message, waiting to write a message, inactive, and stopped due to fault
- The number of available input messages at each processor, represented as a single color-coded spot on the display
- The number of messages queued in the operating system for output to a neighboring node, represented as a color-coded spot
- The traffic (number of messages transmitted between pollings) over each communication channel, coded into the colour of the line segment used to represent the channel.

This information produces a dense, continuously varying display on a 13-inch monitor that we have found gives both an easily comprehended overall status display and enough detail to isolate interesting events. For example, the overall amount and balance of communication activity is perceived through the apparent "hotness" of the communication links, communications bottlenecks show up as noticeable color spots representing backed-up input or output messages, and overall load balancing is apparent from the colors that show whether nodes are computing or waiting.

Our first efforts with programer-planned data have been as follows. The program may contain statements of the form "state(x)" to set a processor's state variable to x which is simply an integral value between 0 an 255, with the purpose of declaring dynamically that the program is entering condition or state x. Although the programer is free to invent arbitrary meanings for these values, we generally expect them to encode the phases or steps of algorithms.

The operating system and the monitor process collect these values along with the other data, and the display presents them along with all the other information described above. (The programer may define encodings and color schemes to be used in displaying these values.) For example, conjugate gradient algorithms consist of iterations of three basic routines: matrix vector multiplies, inner products, and convergence checking (which may be done frequently).

State variables could be used to let the programer know which routine is being executed in each processor, and to show the general timing relationship among the routines.

A further use of state variables would be this: Allow the programer to define, at execution-time, a *target value* for the state variables, where execution on each node is stopped when the node sets its state variable to the target value. Or the programer might specify that execution on all nodes is to be stopped when *any one* of some chosen set of nodes reaches the target value.

By applying a debugger to the stopped program, the programer might interrogate closely for particular values, and then continue execution, perhaps with different targets. This resembles the use of breakpoints in a sequential program, except that it does not rely on the explicit selection of locations in the programs which may vary due to the execution of different blocks of code on different processors.

In the above example, execution could be halted on each processor after the matrix vector multiply was completed so that the programer could check the results by probing with a debugger, or execution could be halted the first time any node reached the convergence checking to see how far along the other nodes were.

These debugging tools, except for the target value idea, have been implemented on our hypercube multiprocessor and we are currently experimenting with them. The software, including supporting software, should be ready for distribution to other sites in early 1988. We would like to obtain feedback from applications programers so that we might evaluate and refine the tools.

"Parallel Algorithms for Solving the Triangular Sylvester Equation on a Hypercube Multiprocessor"

Bo Kagstrom et al., Institute of Information Processing, Sweden.

There are several problems one has to attack when designing algorithms for a multiprocessor architecture. Among these are the choice of granularity of parallelism and the scheduling method of computations. That is, one has to decide the grain-size of the computation to be carried out on each processor and their data-dependencies, how these grains of computation (processes) are to be identified, and how the processes are to be scheduled among the processors. In this talk we consider hypercube architectures with a host, local memory, and message passing, and no shared memory.

Message-passing parallel algorithms are developed for solving the triangular Sylvester equation $AX + XB = C$, where the unknown X is $m \times n$ and A, B, C are given $m \times m$, $n \times n$, and $m \times n$ matrices, respectively, with real entries and A, B are upper triangular. The problem of transforming A and B to upper trianglar form will not be emphasized in this talk.

We discuss the inherent parallelism of the triangular Sylvester equation and introduce the concept of data dependency. In a "coarse grained" view the computation of X can be done in three different ways: X can be computed column-wise, row-wise, or block-wise, leading to three different algorithms.

The sequential algorithms are used in some modified forms to build parallel algorithms. In order to make the algorithms as efficient as possible for different values of m and n and the number of processors, p, we study different partionings and mappings. The matrices A, B and C are partioned by columns, rows, and blocks. The mapping methods used are block and wrap. A theoretical analysis of the parallel algorithms in terms of arithmetic and communication costs is presented.

We provide details in C programs that implement the algorithms on a hypercube simulator and which should run with little modification on real hypercube architectures (e.g., the Intel iPSC). The performance of the various algorithms for different values of m, n and the number of processors p is demonstrated in terms of parallel speedup and efficiency, processor utilization, and communications costs. Results from real hypercube implementations will also be reported (ongoing work).

"Parallel Transonic Flow Calculations"

Choi-Hong Lai, Queen Mary College, UK.

Transonic flow calculations using a domain decomposition technique are discussed and their performance on an array processor is investigated. The emphasis of this paper is on the mapping strategy, namely "sliced mapping," for transonic flow calculations. Advantages of using the sliced mapping for this application are discussed as compared with crinkled mapping.

First, the transonic small perturbation equation is investigated. Particular attention is paid to the various iteration methods within the mixed subsonic and supersonic region. A modified Gauss-Seidel iteration is applied to the above region which shows a good improvement over a parallel AF2 technique. Another technique that has been used is to use a linearised discrete operator of the TSP equation which shows its advantages as compared with the use of the nonlinear discrete operator for low mach number such as 0.7. Second, the technique is generalized to transonic Euler equations.

"A Reconfigurable Multitransputer Network as a Tool for the Experimentation of Parallelism in Scientific Computing"

Pierre Leca and Alain Cosnuau, ONERA, France.

This paper presents the studies done at ONERA using the transputer as a building block for experimentations in the field of scientific multiprocessing.

The architecture of a reconfigurable multiTransputer network, built with standard INMOS modules, is first described and the main hardware features of the building blocks (T800, C004) are given:

Sixteen T800 Transputers, accessing 32 KB of local memory, are connected together in a double ring configuration using two hardware bidirectional links per transputer. The two other bidirectional links are connected to the IMS C004 32-way crossbar switch. The control of this switch is programed via a hardware link of a dedicated T212 Transputer, in such a way that the topology of the 16-transputer network is dynamically reconfigurable by software.

Due to the C004 switch, this system may be, for instance, configured as a linear array, a double ring, a two-dimensional torus, or as more complex networks if each bidirectional link is split into two one-directional links, and may be programed horizontally or vertically using the OCCAM language, whose characteristics are briefly described.

The description of different types of algorithms, with the OCCAM translation and their relative performances, that have been implemented on this multiprocessor is given. It concerns:

(i) Systolic algorithms, 1-D or 2-D systolic algorithms, such as convolution, matrix by vector, or matrix by

matrix products. Moreover, the dynamic switch allows the chaining of two distinct systolic algorithms.

(ii) Coordinating computations, this system may be seen as a set of processors that coordinate their activities by exchanging data via the hardware links. Linear recurrences that appear in "Gauss-Seidel"-like algorithms match perfectly with this architecture.

In the future this multitransputer architecture will be seen as a target of a software tool that automatically detects a potential parallelism between FORTRAN instructions at loops level. We will give some indications about the possibility to automatically generate a parallel code and a network configuration for this architecture.

"Nested Dissection Orderings for Parallel Sparse Cholesky Factorization"

John Lewis, Boeing Computer Services Co., US.

The time required to factor a sparse symmetric matrix on a parallel computer is strongly affected by symmetric reorderings of the matrix. The data dependencies that restrict parallelism in the factorization are given by the elimination tree induced by the ordering. Thus, one standard measure of (and a lower bound on) parallel completion time is the depth of the elimination tree. In previous work – U1 – we have found nested dissection orderings for general sparse matrices that appear to be better orderings for parallel factorization than the best parallel versions of the best standard sequential orderings. The nested dissection orderings, have less deep elimination trees than the parallelized sequential order-ings, and their sequential measures of complexity are essentially equivalent to those of the sequential orderings.

The orderings in U1 are based on the Fiduccia-Mattheyses graph partioning heuristic. We have observed that this heuristic is quite sensitive to a number of implementation parameters, making it difficult to create a generally effective ordering algorithm. In addition the graph partitioning heuristic is sequential and is quite expensive. In this work we explore several alternative approaches to finding nested dissection orderings of similar quality. Our goal is a cheaper and more robust ordering method.

Our approaches include:

- Parallel implementation of variants of the Kernighan-Lin and Fiduccia-Mattheyses graph partioning algorithms
- Graph partioning by simulated annealing
- Variants of the George & Liu automatic nested dissection procedure.

The performance of these ordering heuristics is compared on a set of sparse matrices obtained from actual applications.

"Applying a Sequence of Plane Rotations on a Vector-processing Machine"

Jeremy Du Croz et al., Numerical Algorithms Group Ltd., UK.

When eigenvalues and eigenvectors, or singular values and singular vectors, are computed by means of the QR algorithm, a substantial amount of time is spent in applying sequences on plane rotations to the rows or columns of a matrix. Indeed, if we consider just the last step in computing the singular value decomposition, namely computing the singular values and vectors of a bidiagonal matrix, the computing time is dominated by the time spent in applying plane rotations to update the matrices of left and right singular, it is therefore desirable to perform the plane rotations as efficiently as possible.

The Level 1 BLAS routine, SROT, can be faster than in-line FORTRAN code, but, especially on a machine with vector-registers, it gives only a limited speedup because it takes no advantage of the chaining between successive rotations. Much better performance can in principle be obtained by a routine of larger granularity (similar to that of the Level 2 BLAS) which applies sequences of plane rotations. Such a routine has been included in Mark 13 of the NAG Library. It can apply sequences of plane rotations to either the rows or columns of a rectangular matrix in any of the orders

(1,2),(1,3),(1,4),...,(1,n)
(1,n),(2,n),(3,n),...,(n-1,n)
(1,2),(2,3),(3,4),...,(n-1,n)
(n-1,n),...,(3,4),(2,3),(1,2)

In order to achieve optimal use of vector-registers, it is necessary to code the routine in assembly language. We shall demonstrate the speed obtainable with such an assembly language version, and the speedup which it confers on NAG Library routines for computing eigenvectors and singular vectors.

We shall propose a specification for the routine, and recommend that efficient implementations be provided by manufactures in the same spirit as for the BLAS.

"Nonlinear Transport Calculations in 1-D MOSFETs Using a CRAY X-MP/48 and a Sequent Balance Multiprocessor"

J.A. McInnes and S.A. Mogstad, University of Strathclyde, UK.

The electrical conductance in a disordered system may be stated in terms of a set of nonlinear equations. In macroscopic systems the nonlinear effects are often negligible or self-averaging and the appropriate equations linearised. In the case of submicron 1-D MOSFET's nonlinear effects must be retained in the formalism.

Algorithms for solving the correspondence nonlinear equations are developed on both a CRAY X-MP/48 and a Sequent Balance multiprocessor. The computed magnetic field dependence of electron transport in submicron MOSFET's for varying applied gate voltage is compared with experimental observations.

"Supercomputing in Denmark"

Bjarne Stig Anderson et al., The Supercomputer Group, The Danish Computing Centre for Research and Education, Denmark.

Three major university computing centres, RECKU (Copenhagen), RECAU (Aarhus), and NEUCC (Lyngby) have united to form one supercomputer centre for Denmark, called in English the Danish Computing Centre for Research and Education, abbreviated to UNI*C.

A major event for the Danish scientific community was the recent installation of the Amdahl VP 1100 at UNI*C. This supercomputer is intended for joint technological research carried out by the Danish universities, research institutes, and industrial companies. Besides the Amdahl VP UNI*C has IBM 3081, Sperry 1100/92, CDC Cyber, Alliant, and several VAXes.

The Amdahl VP 1100 has one CPU with a peak performance of 286 Mflop/s. The vector and scalar instructions can run simultaneously. The 30 Gbyte disk storage is supplemented with IBM 3420 tape drives. This is in addition to the large core of 128 Mbytes. The lessons of the first year will be described, including experience with the national network being established to connect all university users in Denmark to the computer. The network uses a 2-Mbit/s trunk.

The NAG library has been implemented and specialized, and the basic sub-routines (BLAS) have been adapted for the Amdahl VP. Many user programs have been converted, and these have demonstrated the excellent performance of the vectorizing FORTRAN compiler. Also, the very large memory, 128 Mbytes, contributes to much shorter program elapse times, e.g., by moving temporary data sets into main memory. Examples will be given and explained.

"Cycle Reduction and Matrices with a Group Structure"

Hans Munthe-Kaas, Norwegian Institute of Technology Section for Numerical Mathematics.

Cyclic reduction is a variant of Gaussian elimination for solving special systems of linear equations. It has been successfully applied in fast Poison solvers, in direct parallel methods for solving tridiagonal equations and as a parallel preconditioner for iterative methods.

In the talk we will show that cyclic reduction may be described in the language of group connected with the matrix graph. This may be done in many different ways, giving rise to different algorithms. We investigate generalizations of the standard algorithms and show that the generalized algorithms perform better than the standard ones in solving simple tridiagonal systems on Cray X-MP. Furthermore we will implement and compare different parallel preconditioners for iterative methods, based on cyclic reduction-like schemes.

"Evolution Algorithms in Combinatorial Optimization"

H. Muhlenbein et al., Gesellschaft für Mathematik und Datenverarbeitung mbH, West Germany.

Evolution algorithms for combinatorial optimization were proposed in the 1970's. They did not have a major influence. With the availability of parallel computers, these algorithms will become more important.

In this paper we discuss the dynamics of three different classes of evolution algorithms: network algorithms derived from the replicator equation, Darwinian algorithms, and genetic algorithms inheriting genetic information.

We present a new genetic algorithm which relies on intelligent evolution of individuals. With this algorithm, we have computed the best solution of the famous traveling salesman problem. The algorithm is inherently parallel and shows a superlinear speedup in multiprocessor systems.

"Data Distribution and Communication for Parallel Analysis of 3-D Body-Scan Data"

M.G. Norman et al., University of Edinburgh, UK.

It is widely anticipated that the flexibility of MIMD architectures will allow efficient parallel implementation of middle- and high-level vision applications. In truth, the problems of achieving high levels of processor utilization while keeping communications overheads low, mean that middle- and high-level vision applications can only be implemented efficiently on MIMD parallel architectures with great difficulty.

This paper describes an attempt to implement low- and middle-level vision algorithms on the MIMD architecture of the Meiko Computing Surface. The image being processed was the three-dimensional data produced by medical magnetic resonance imaging. The size of the dataset meant that it was not possible to store it on a single processor, and it was distributed among processors in a way that attempted to maximize the efficiency of a middle-level algorithm – tracking of surfaces within three-dimensional datasets – while maintaining the efficient implementation of low-level operations.

This paper also describes the details of the finished system including the necessary communications harness, a generalization of the Zucker-Hummel surface detection operator to non-square metric, and three-dimensional surface display.

"Diffusion Limited Aggregation – Model and Methods"

M.A. Novotny et al., Bergen Scientific Centre IBM, Norway.

The Diffusion Limited Aggregation (DLA) model introduced in 1981 has been used to model a wide variety of physical phenomena, including viscous fingering in Hele-Shaw cells, fluid-fluid displacement in porous media, electric breakdown, electrodeposition, the

formation of snowflakes, and irreversible kinetic aggregation.

The underlying equation of this model is the Laplace equation, $\Delta^2\phi = 0$, subject to the boundary conditions that $\phi = 1$ at infinity and $\phi = 0$ on boundary of the aggregate. The boundary moves as the aggregation process proceeds – producing fractal aggregates. We will discuss various modifications of the original DLA algorithm which allow the inclusion of additional physical processes such as surface tension. In addition, we will describe the concept of noise reduction and show that contrary to the accepted current philosophy, noise reduction changes the asymptotic shape of DLA aggregates.

We have implemented this algorithm on a six-processor IBM 3090 using Parallel FORTRAN, and will describe how this inherently scalar algorithm may be parallelized and vectorized.

"One-Way Dissection with Pivoting on the Hypercube"

T.-H. Olesen and J.R. Gilbert, Chr. Michelsen Institute, Norway.

A one-way dissection ordering is used for Gaussian elimination with partial pivoting to solve 2-D elliptic finite difference and finite element problems on a hypercube parallel processor. These problems can be put into a matrix-vector form, $Ax = f$, where the matrix A takes the place of the differential operator, x is the solution vector, and f is the source vector. Using "normal width" separators creates problems while triangularizing the A in parallel; an inordinate amount of communication is needed, which destroys the benefits of using a parallel computer.

One solution to this communication problem, which is used in this report, is to use double-width separators. With this type of dissection, the domain is spread across all the nodes without any overlap. Each processor is given its own part of the source vector and computes its own part of the stiffness matrix, A.

The elimination starts out in parallel; communications is only needed after most of the elimination is finished when the separators need to be eliminated. Back substitution is initially done on the separators, and then totally in parallel without communication on each node. This method extends the size of the linear equations that can be solved directly on the hypercube.

"A Quadratically Convergent Parallel Eigenvalue Algorithm Based on Jacobi-Like Transformations"

M.H.C. Paardekooper, Tilburg University, the Netherlands.

This paper discusses a generalization of the Jacobi process (1846) for arbitrary almost diagonal $n \times n$ matrices, n even. In each step of the final stage 1/2 n symmetrically placed pairs of nondiagonal elements are annihilated. We prove that with the caterpillar pivot strategy the sequence of transformed matrices converges to diagonal matrix. The quadratically convergent method, also in case of multiple eigenvalues is appropriate parallel computation on an array processor.

Proceeding to the annihilation process with its local information structure a parallel norm-reducing Jacobi-like process transforms the initial matrix into an almost diagonal one, adapted to the final, quadratically convergent annihilation method. In this pretreatment each step in a direct sum of unimodular shears. As a consequence of the invariance of the Frobenius matrix norm under orthogonal transformation the norm reduction should be discussed in theoretical terms that are invariant under these transformations. In our modification of Sameh's algorithm (1971) the problem formulation with so-called Euclidean parameters inproves and simplifies the procedure and the formulae in the related minimization problem.

Different aspects of the method and its successful implementation (Communication, storage strategy) on The Delft Parallel Processor will be discussed.

"The Timing of Sort Algorithms on the Amdahl 1200 Vector Processor"

Bodo Barady, Vector Processor Technical Services, California, USA and Kenichi Miura, Mainframe Division of Fujitsu Ltd., Japan.

Methods for sorting integers into ascending order on the Amdahl 1200 are compared. Algorithms have been modified to increase the performance of sorting integer arrays and to provide the sorting of arrays using a pointer list. A variant of the radix sort, in the limit of a binary radix, can be fully vectorized on the Amdahl Vector Processor; it sorts 12,800 elements in less than 5 milliseconds. Its performance is compared to widely used sorting methods.

"Problem Parallelism Versus Processor Parallelism"

D. Parkinson, Queen Mary College, UK, and R. Francis, La Trobe University, Australia.

The literature of parallel algorithms is full of papers which discuss the maximal parallelism in tasks. In practice, real multiprocessor systems have fewer processors than demanded by the algorithm.

A typical example occurs in image processing algorithms on n x n images with n = 512 or larger. Processor arrays of this size only exist in imagination and so the practical problem is how to properly use a smaller array.

The experience gained in implementing an image-component-labelling algorithm for 512 x 512 images on 32 x 32 DAP's will be presented. Relative timings of a variety of strategies will be presented.

"FORTRAN as a Parallel Programing Language"

Ph. D. John R. Perry, Alliant Computer Systems Corp., US.

This is an overview of an approach in computer and compiler architecture that allows parallel programing

within the standard FORTRAN-77 language. This approach provides the capability to exploit implicit fine-grained parallelism of loop iterations automatically, as well as course-grained parallelism at the subroutine or subroutine tree level under the more explicit control of the programer. These capabilities are implemented in the Alliant FX/8 and FX/4 mini-supercomputers and the FX/FORTRAN compiler.

In this paper, we discuss solutions for overcoming the various technical barriers to achieving parallel processing within the bounds of the standard FORTRAN language. These solutions include hardware features to support efficient management of parallel computations, and compiler capabilities in such areas as dependency analysis, memory management for parallel computations, interprocedural analysis, and idiom recognition. We present various examples of FORTRAN scientific programs to illustrate each of these capabilities, and performance results to show the parallel speedup of various classes of problems.

Finally, we discuss ongoing efforts to evolve FORTRAN into a parallel processing language. In particular, we show how features of the proposed FORTRAN-8X standard contribute to the ability to exploit implicit parallelism, and we survey the efforts underway to define a standard for extending FORTRAN with explicit parallel processing constructs.

"Optimal Absorbing Boundary Operators"

B. Rosland and J. Petersen, Bergen Scientific Centre IBM, Norway.

When computing solutions to the two-dimensional wave equation in unbounded domains using finite difference discretization, an artificial boundary is introduced. A boundary condition which absorbs all outward propagating waves must then be used. Equivalently, the finite difference operator must be replaced at the boundary with an appropriate boundary operator.

Approximations to boundary operators have been published. These work well for waves which are propagating towards the boundary near normal incidence. However, problems occur in cases where sources are far from the center of the model or if the velocity field is not homogeneous. In such cases one tends to get results which are contaminated with noise scattered back from the artificial boundaries.

We propose a nonlinear least squares method for determining an absorbing boundary operator. The operator is chosen by demanding that waves traveling within a predetermined cone are attenuated as much as possible. We solve the problem with a Monte Carlo-type minimization method.

We show how the improvements in the results can be used to reduce the storage requirements by reducing the grid size and will show results from the vectorization of the method.

"A Divide and Conquer Method for the Orthogonal Eigenproblem"

W.B. Gragg, Naval Postgraduate School, California, and L. Reichel, Bergen Scientific Centre IBM, Bergen, Norway.

In divide and conquer methods for eigenproblems, the original eigenproblem is split into several smaller subproblems by using rank on modifications. The subproblems are then solved independently, and from their solutions the solution of the original eigenproblem can be determined. This approach has been developed for symmetric matrices by Cuppen, Dongarra, and Sorsensen and implemented on a parallel computer by Dongarra and Sorensen.

Their implementation demonstrates that divide and conquer methods are very suitable for a parallel computer. We will describe a divide and conquer method for the computation of eigenvalues and eigenvectors of orthogonal matrices. Among the applications of the orthogonal eigenproblem are the computation of Pisarenko frequency estimates in signal processing, and the generation of Gaussian equadrature rules for the unit circle.

"Continuation of Parameter-Dependent Partial Differential Systems on a Hypercube"

D. Roose et al. Belgium.

We present a parallel algorithm for the continuation of finite-difference discretizations of elliptic differential systems on a hypercube concurrent processor.

Continuation procedures are used for the analysis of a nonlinear problem depending on a parameter – i.e., for the computation of a branch of solutions in a function of the parameter. In general, each continuation step consists of a predictor and a corrector part. The parallel algorithm is based on the continuation code PITCON of Rheinboldt and Burkardt. In this case a predictor is chosen along the tangent to the branch and a Chord-Newton iteration is used for the corrector part.

Most of the computing time during both the predictor and the corrector steps is spent in the solution of linear systems with a matrix representing the linearization of the original problem. For a finite-difference discretization of a two-point boundary value problem, this matrix is a bandmatrix bordered with an additional row and column.

An efficient solution of the linear systems is obtained by combining the block-elimination algorithm of Keller and the Parallel band solver proposed by Saad and Schultz. For discretizations of a two-dimensional elliptic problem, systems have to be solved with a border block tridiagonal matrix. In this case an algorithm related to domain decomposition is used. The distribution of the data among the nodes is imposed by the solution procedures for the linear systems. We show how the remaining parts of the continuation algorithm can be organized efficiently, taking into account the particular distributions. Timing results for the iPSC hypercube will be presented.

"The Impact of the IBM 3090 Vector Facility on the Data Analysis for JET, The Major European Nuclear Fusion Project"

R.T. Ross et al., JET Joint Undertaking and IBM UK.

JET, the Joint European Torus, is the largest single project of the nuclear fusion research program of EURATOM; The objective is study of plasmas in conditions approaching those required for a fusion reactor. The experiment is run in pulsed mode every 15 minutes and each pulse yields some 12 Mbytes of experimental data. These data are sent to the recently installed IBM 3090-200E with Vector Facility (VF) and a series of data analysis, database storage, and display programs are run in order to give a detailed analysis of the pulse within the 15-minute interpulse period to provide essential information for the subsequent pulse.

This paper shows where the use of the VF has made a significant impact on the level of analysis possible within the interpulse period. One program, IDENTC, which is particularly CPU-intensive, is essential for an accurate determination of the plasma boundary and the internal geometry of the plasma. It achieves this by solving the Grad-Shafranor equation using finite element techniques. The various methods used to optimize this program to take full advantage of the VF are discussed.

These include:

- FORTRAN code restructuring to make maximum use of the multiple/add compound vector operation
- The writing of a key subroutine in IBM Vector Assembler language.

"Design Aspects of a Linear Algebra Package for the SUPRENUM Machines"

W. Ronsch and H. Strauss, Professor Dr. Feilmeier, Junker & Co., GmbH, West Germany.

The SUPRENUM multiprocessor computer system is designed to be a distributed memory message passing (dmmp-) machine expected to reach a peak performance of nearly 4 Gigaflop/s. Each of the 256 processors has a scalar as well as a vector unit and can be controlled by its own program, although many applications will use the mode of single program multiple data programing.

The development of a linear algebra package for this machines has to take into account several aspects; among them:

- The communication costs – i.e., the ratio of computation speed to communication speed
- The mapping strategy for the data objects
- The mapping consistency of the data objects throughout the whole package
- The load balancing of the processors used in the computation
- The numerical stability of the algorithms chosen
- The overall performance of the package – i.e., the efficiency of each routine.

In this paper we present our methodology for data partitioning, the storage scheme chosen for the data objects (matrices, vectors, and scalars), and the scope of the linear algebra package. It is our plan to hide the communication from the user and to give the parallel routines a calling sequence which is, as far as possible, in accordance with the corresponding LINPACK and BLAS routines.

In designing the package some emphasis was also laid on the easy portability of the package to other dmmp-machines such as the Intel iPSC and the Floating Point T-Series.

The package is being implemented in SUPRENUM FORTRAN which is an extension of FORTRAN 77 with some 8X-features and communication instructions for message passing.

Some first results from our experience of the SUPRENUM preprototype are given.

"Implementation of Pre-Stack Depth Migration on IBM 3090"

Ottar A. Sandvin and Torstein Egilson, Norway.

In seismic processing it is of utmost importance to achieve optimum spatial resolution of the subsurface. In particular, this is important in complex geological structure where the potential reservoirs are most likely to be located. Seismic migration is the processor that over the last years has proven most successful in producing an unbiased image of the subsurface. However, the full potential of the migration process is not fully exploited in routine seismic processing as the migration is usually done at the end of the processing sequence.

Hence we want to demonstrate a migration scheme that is done pre-stack and on a shot level in order to remove the smoothing effects introduced by previous processors. Pre-stack migration on single shots makes the process very suitable for concurrent processing as individual shots can be processed separately on each cpu. The three main features of the process can be summarized as:

1) Forward propagation of source field to given depth level

2) Backward propagation of receiver wave field

and finally

3) The imaging process at the same level

All the three processes can be shown to be accomplished through spatial convolution, which makes them very attractive for vectorization.

The same operations are repeated for every frequency, which also introduces another level of concurrent processing. Storage requirements, concurrent processing, and vectorization will be demonstrated during the presentation. Also, some numerical examples on realis-

tic synthetic data will be presented as a demonstration of the imaging potential of the method.

"High Resolution Numerical Simulations of Incompressible Turbulent Flows on the IBM 3090 Vector Multiprocessor"

P. Santangelo et al., IBM ECSEC, Italy, and B. Legras, Laboratoire de Meteorologie Dynamique, France

In recent years many numerical and theoretical studies have been performed to understand the dynamical properties of turbulent flows. The present available vector multiprocessors allow for a direct numerical integration of the incompressible Navier-Stokes equations in two dimensions and with very high resolutions (1024^2). Besides theoretical motivations, 2-D turbulence is of interest to geophysical fluid dynamics, as well as astrophysics and plasma physics.

The main feature of 2-D turbulence is the appearance of coherent structures inside the flow; these structures are extremely stable and their motion dominates the flow, suggesting the idea that only a few degrees of freedom would be necessary to characterize the fully turbulent motion. This scenario has been found to be consistent with the results of many numerical simulations, independently of the initial conditions and both for decaying and forced turbulence.

"Turbulent Air Flow in Disk Files"

M. Briscolini et al. IBM ECSEC, Italy.

In this paper we are concerned with numerical simulation of the turbulent air flow inside the magnetic disk storage systems. Using the IBM 3090/VF, we have investigated the flow pattern that arises between and around the rotating disks at high speed in an enclosed space. We use a pseudospectral formulation, and in considering the geometry of the system we force directly the boundary conditions of the velocity field. This new computational technique, introduced by Basdevant and Sadourny, was later extended by Briscolini and Santangelo. We have examined this problem in the two-dimensional approximation and in the three-dimensional case, and we present here a comparison between these calculations.

"The IBM Parallel FORTRAN Language"

J.F. Shaw et al., IBM, US.

The IBM Parallel FORTRAN language offers facilities for small- through large-grain parallel execution. It provides automatic parallelization and vectorization of loops, explicit language for specification of parallel work, and permits multiple levels of parallel executions. The language is designed to run on high-performance, shared memory multiprocessors, where dedicated processors may not be appropriate, and where the number of processors may change from moment to moment.

This paper will describe the design of the language, discussing the model of programing it implements. The language allows a programer to distribute parallel work to processors, rather than have processors coordinate on a single piece of work. The language stresses data integrity and isolation by providing explicit dynamics control over the sharing and copying of common blocks. Finally, Parallel FORTRAN is intended to be a language that is independent of the machine configuration and the operating system.

"Digital Reconstruction of Images from Their Projections Using a Parallel Computer"

Fiorella Sgallari et al., Universita di Bologna, Universita di Firenze and CINECA Computing Center, ITALY.

The problem of image reconstruction from projection has arisen independently in a large number of scientific fields. Of all the applications, probably the greatest effect on the world at large has been in the area of diagnostic medicine: computerized tomography (CT).

In the last decade a good number of algorithms and computational techniques dealing with the reconstruction problem have been presented. The mathematical essence of the procedure is the estimation and presentation of a real-value function of several variables from approximate values of a finite number of its line integrals (the so-called X-ray or Radon transform).

As far as the numerical reconstruction methods are concerned, we have a considerable number of algorithms, generally unrelated with each other. The reason perhaps can be found in the intrinsic ill-posedness (in the sense of Tikhonov) of the reconstruction problem: the fact makes practically useless the analytical inversion formula for the Radon transform.

We believe that the well-known ART and SIRT reconstruction methods present some remarkable points of interest. From a purely conceptual point of view, they can be placed in the general framework of the regularization techniques based on Tikhonov's "smoothing functional;" of course, further generalizations of the existing methods can be devised by choosing different "smoothing functionals."

Conversely, solution techniques based on relaxation algorithms can be easily implemented on parallel computers, exploiting their specialized arithmetic units. It is worth noting that the convergency of such iterative algorithm have been deeply investigated in the recent literature, taking into account the relevant fact that the updating process is in principle completely asynchronous (some authors define it a "chaotic iteration process").

In this paper, after some theoretical remarks, an example of "smoothing functional" is provided. The convergence of the related parallel iterative procedure is proved. The implementation on a CRAY-X/MP 48 computer is described; experimental results and comparisons with synchronous algorithm are also reported.

"Trends in Supercomputing"

Karl-Einar Sjødin, Control Data A.B., Sweden.

The ETA 10 supercomputer family includes the most powerful supercomputers of today. It also has features and functionality that can be extrapolated into industry trends.

1. Architecture and technology:

- Structure of processors and memories
- Technology roadmaps for logic memory and communication/networking.

2. Software - operating system and programing issues:

- UNIX (or rather POSIX) is from now on the standard for supercomputer operating systems
- FORTRAN will continue to be the standard programing language for years to come
- Multitasking using parallel processors is of rapidly growing importance.

3. Uses of supercomputers:

- Established and new application areas
- Departmental supercomputing
- Supercomputers and workstations
- Simulations and visualizations

Do we really need the teraflops/terawords supercomputer?

"Concurrent Dynamic Simulation of Distillation Columns via Waveform Relaxation"

Anthony Skjellum and Manfred Morari, California Institute of Technology and Sven Mattisson, Lund Institute of Technology, Sweden.

The distillation column and its variants are arguably the most important class of unit operations in chemical plants. Dynamic models consist of differential-algebraic systems with stiff, nonlinear ordinary differential equations modeling fluid flow and mass balances, and nonlinear algebraic equations modeling the vapor-liquid equilibrium. A multicomponent system with a practical number of columns can have on the order of 1000 equations or more. Furthermore, physical property calculations for vapor-liquid equilibrium can be outstandingly arduous. Hence, the dynamic simulation of distillation columns continues to receive serious attention because detailed models often have huge computing requirements. It is therefore natural to seek concurrent algorithms suitable for large-scale concurrent computers in order to obtain significant simulation speedup.

In this paper, we take the first steps toward reaching this goal. We consider a highly simplified binary distillation model with simple vapor-liquid equilibrium. A set of stiff, coupled nonlinear ordinary differential equations result. Importantly, these simplications still allow us to capture three central issues: stiffness, large problem size, and computational difficulties arising from systems which incorporate coupled columns. Waveform relaxation was chosen as the solution approach because it offers large speedup potential. For example, temporal latency of subsystems can be exploited, implying reduced computational effort for such subsystems and, correspondingly, the opportunity for greater overall speedup in the total simulation.

Because of the success of the Mattisson's CONCISE simulator for VLSI circuit simulation via waveform relaxation, we have formulated the distillation model within CONCISE. We present preliminary speedup results obtained with CONCISE on a binary n-cube computer system. We also indicate how the use of more demanding models (including, for example, complex vapor-liquid equilibrium) is likely to change the observed concurrent efficiency. Future directions of this research are also indicated.

"A Formal Model and an Empirical Metric for Memory Latency in Multiprocessors"

David F. Snelling, University of Leicester, UK.

All multiprocessors, particularly shared memory multiprocessors, are affected by memory latency. Usually there is some performance loss as memory paths become longer, or bank conflicts and synchronization delays increase.

The classical techniques of compensating for memory latency, such as vectorization and instruction pipelining, are effective in serial processing environments, but to little to alleviate the performance degradation due to multiprocessing. In this paper the fundamental aspects of memory latency are identified and outlined, and a generalized, formal model for memory latency is proposed.

This formal model is compared to an empirical metric for determining the actual effect of memory latency on a particular multiprocessor running a given algorithm. This memory latency metric is a measurable quantity that indicates the extent to which a particular multiprocessor architecture can hide or mask the memory and synchronization latency inherent in a given algorithm.

The ability of several multiprocessors to mask latency is analyzed using these two concepts. The Denelcor HEP and the transputer-based, shared memory multiprocessor being developed at the University of Leicester will serve as case studies. These two machines are known to have the ability to hide memory and synchronization latency and are therefore highly suitable as case studies. Nonlatency-hiding machines will also be discussed.

"Implementing and Tuning Multigrid on Local Memory Parallel Computers"

Karl Solchenbach, SUPRENUM GmbH, West Germany.

Multigrid (MG) methods are often claimed to be not suitable for highly or massively parallel multiprocessor systems with many processors because of the "lack of parallelism" on the coarse grids. Indeed, if implemented in

a straightforward manner the multiprocessor-efficiency of MG methods on local memory machines may be significantly lower compared to single grid methods. The degree of efficiency reduction is determined on one hand by the system characteristics (e.g. time for communication) and on the other hand by the implementation of the coarse grid data structure on the multiprocessor system.

The purpose of this paper is firstly to investigate several possible coarse grid techniques and second to show how the efficiency of a given MG-algorithm depends on critical system parameters. As a result of both considerations we expect some knowledge on how to tune parallel MG methods optimally to a given multiprocessor system by proper choice of the coarse grid techniques. The basic requirements for an implementation of MG on parallel systems are briefly outlined, namely,

- MG components with a high degree of inherent parallelism (like red-black relaxation)
- Data distribution concept that ensures optimal load balancing and minimal communication (grid partitioning).

In order to have a flexible tool which allows performance estimates of different coarse grid techniques and different multiprocessor systems, an MG simulation program was developed. Its underlying assumptions, its calculation and communication model, and its facilities are described. The aim is to evaluate different techniques for the treatment of the coarse grids on local memory machines, to analyze the dependence on system parameters, and to predict the behavior of these techniques on parallel machines which are not available yet (like Suprenum). Different possibilities of accelerating the coarse grid calculations on local memory machines are:

- Stop the coarsening process and perform relaxations on some intermediate level
- Collect the coarse grid data to fewer processors
- Use larger overlap areas in order to reduce the number of messages.

A detailed analysis of the calculation and communication work related to these techniques is given. Finally, results of the simulation are presented. The estimated data are compared to measurements on existing local memory machines. Performance estimates of the different coarse grid techniques are given. The multiprocessor-efficiency of the overall MG method is estimated and requirements on the multiprocessor system characteristics are derived.

"Efficient Parallel Implementable Algorithms for Determination of Line-of-Sight Visibility"

Arun K. Somani, University of Washington, Seattle, and Farid Mamaghani, BBN Dalta Graphics, Inc., US.

This paper presents two efficient and fast algorithms which vectorized the task of finding the visible points within a square of side 2n (or a circle of radius n) on a plan view display of terrain maps from any given viewpoint. These algorithms either can be implemented on a sequential computer or on special-purpose parallel hardware.

Inadequacies and inefficiencies of some of the techniques currently used to achieve this are discussed. The details of a pipelined parallel architecture for implementation of the new algorithms is also presented.

"Divide and Conquer Algorithms for SIMD Architectures"

Lennart Johnsson, Thinking Machines Corporation, US, and D.C. Sorensen, Argonne National Laboratory, US.

Divide and conquer algorithms for the symmetric algebraic eigenvalue problem have been successful on both shared memory and distributed memory architectures.

We investigate the suitability of this scheme for SIMD architectures such as the Connection Machines. A version of the algorithm that is suitable for computing eigenvalues of a symmetric tridiagonal matrix will be presented.

This algorithm has potential for utilizing all processors of a large SIMD array throughout the course of finding all the eigenvalues of the given matrix.

"Parallel Multigrid for Solving the Steady-State, Incompressible Navier-Stokes Equations on General 2-D Domains"

Dr. Klaus Stuben, Gesellschaft für Mathematik und Datenverarbeitung mbH, West Germany.

Multigrid methods for solving elliptic partial differential equations are known to be optimally efficient on sequential computers – i.e., the computational work required to solve the corresponding discrete system of equations is proportional to the number of unknowns. The additional fact that all components of a multigrid method are fully parallelizable, make such methods highly attractive for parallel computers.

A parallel (MIMD, local memory) multigrid method for solving the steady-state Navier-Stokes equations on general 2D-domains is presented, based on a finite-volume discretization for general (nonorthogonal) meshes. The method served as the basis of a program package for the parallel solution of general 2-D flows. Details on the multigrid algorithm as well as on the general program design will be given.

Although the program is actually being developed in the German supercomputer project SUPRENUM, high priority has been given to the requirement of portability: The program is designed so that it can be used both on sequential and parallel computers. This is achieved by means of a flexible and general library of "communication routines": all machine-dependent constructs are completely hidden inside these routines, which perform the mapping of processing as well as the communication.

"Lattice Gas Simulations of Two-Dimensional Turbulence on IBM 3090/VF"

S. Succi and P. Santangelo, IBM/ECSEC, Italy.

Lattice Gas (LG) models obeying cellular automata (CA) rules have recently gained a great deal of interest as a new alternative tool for the simulation of complex hydrodynamic phenomena. In fact, despite of the fully discrete nature of their underlying phase space, LG do exhibit some physical features like sound waves, viscous behavior, and hydrodynamic instabilities that are typical of continuous systems such as fluids. From the computational point of view, the essential property of CA is the fact of working with quantized variables – i.e., variables which take values only in a small set of integers, typically just zero or one, and are consequently represented exactly with only a few bits per dynamic variable in the computer storage.

In this paper, we discuss the few and simple criteria which govern the efficient implementation of CA rules on the IBM 3090 vector multiprocessor. Performance data are also presented and commented on. As a physical application, we present some results concerning the free decay of halogenous turbulence in two dimensions, obtained running the hexagonal Frisc-Hasslacher-Pomeau lattice gas on a very-high-resolution grid containing 8192x8192 nodes. One of the main results emerging from these high-resolution runs is the neat detection of the long-range forces governing the interaction between the coherent structures (vortices) present in the fluid.

"Numerical Software Development for Local Memory Machines"

Bernhard Thomas, SUPRENUM GnbH, West Germany.

Scientific supercomputing on large-scale parallel machines encounters essentially new problems – as compared to vector supercomputers – both with respect to development and performance of programs. As for program development, handwriting efficient parallel code anew still takes some effort, whereas automatic parallelization of existing code is only just at its beginning. As for performance, well-established sequential algorithms may not, or not efficiently, be parallelized, and new parallel algorithms have to deal with possible lack of stability. However, there are problem areas and machine architectures where some of these problems can be substantially relaxed.

In the SUPRENUM project, a parallel supercomputer is developed to deliver performance in the range of several Gflops for scientific computing. Basic components of the local memory architecture are powerful vector processing nodes, connected by a very fast hierarchical bus system. Concurrently with the system development, software developments in SUPRENUM attack the above problems on different levels:

- Parallel numerical software is developed that optimally exploits the system's capacities, e.g., basic routines in linear algebra, multigrid PDE-solvers, grid generation, and CFD packages.
- Libraries are provided that facilitate writing of portable and efficient parallel programs by means of standardized, optimal routines for typical tasks arising with message passing systems, e.g., generations of and communication within a process system, mapping to the target hardware.
- Tools are developed for interactive parallelization of existing code and for the support of generating new parallel applications programs.

This paper gives details about these strategies, with an emphasis on software developments for grid-based problems in conjunction with the message passing library.

"Parallel Processing Within a Virtual Machine"

L.J. Toomey et al, IBM Kingston, US.

The ability of single scientific applications to utilize the full computational capability of a tightly coupled, shared memory, multiprocessor system is becoming increasingly important. To date, applications run with CMS in a virtual machine have been restricted to a uniprocessor view. This paper describes a method of enabling numerically intensive applications to simultaneously use all processors of a multiprocessor configuration within a single virtual machine.

Virtual CPU's, a feature of VM/XA SF and VM/XA SP1, are used to specify dispatchable units of work to the VM control program. A CMS server is described that allows all virtual CPU's to obtain CMS services. The methods discussed have been implemented for FORTRAN applications. However, these methods are applicable to other languages as well. The implementaiton is the Basis for the IBM Parallel FORTRAN PRPQ on VM/XA SP1.

"Implementational Aspects of ADA for Vector Processing Target Machines"

Peter Wehrum et al., Siemens AG and University of Marburg, West Germany.

The purpose of this paper is to describe some aspects of parts of the MAP project 786, "Parallel Numerical Processing through Ada" (PNPA). This project has been dealt with by Siemens AG together with the University of Marburg. The aim is to examine the programing language Ada for its capability in the field of scientific parallel programing; i.e., to look to the problem of how to write efficient numerical applications for supercomputers in Ada, and how to translate them into highly parallel machines code. This part of the work is centered on vector processors and vectorization techniques.

Ada may be regarded as a programing language appropriate for the writing of scientific and, in particular, parallel applications. First of all, Ada supports software engineering concepts such as programing in the large, software reuse, reliability, and readability. Language fea-

tures supporting these concepts are, inter alia, separate compilation, generic subprogram and packages, exception handling, overloading, and operator definition.

Regarding the granularity of concurrent language constructs, parallelism may occur on at least three different levels:

- Expression level
- Statement level
- Module level.

The latter is supported by Ada through tasking; the former two levels, however, are not directly covered by Ada language features. As for parallelism at the expression level, Ada provides only a few predefined operations for arrays but does not supply predefined arithmetic vector operations.

Basically, there are two approaches which make it possible to exploit the facilities offered by a vector processor:

First, an auto-vectorizing compiler is built and utilized for the translation of scientific programs, which may be written in sequential, scalar Ada. The algorithms in this kind of compiler may be based on vectorizing techniques as described, for example, by Kennedy et al., Kunk et al. and Lamport.

Second, a standard generic package is provided that specifies vector and matrix operations. Instantiations of this vector package yield the appropriate operations in high-level vector notation, on which scientific applications can then be based. The bodies of the vector operations may be implemented by means of low-level features, such as machine code insertions and interfaced Assembler and FORTRAN subprograms.

In the PNPA project, both approaches have been followed in that a vector package has been specified (which is simpler and less expensive to implement), and the requirements of an auto-vectorizing compiler have been studied.

Additional requirements discussed within the project concern the environment in which computer aided software engineering (CASE) for vectorizing Ada programs proceeds, *including an interactive vectorizing optimizer* for preprocessing Ada code, and further APSE elements such as a debugging tool and a vectorized numeric library.

The interactive vectorizing optimizer performs a code-to-code transformation at source level supporting vectorization in order to ease the vector analysis process of later compilations. This tool has been specified by graph-related techniques as a knowledge-based system including a learning facility.

The environment is based on a workstation network connected to a scalar front-end and the vector back-end processor. The vector target machine primarily concentrated on is the Siemens 7.800 VP 200 / Fujitsu VP 200.

"Parallel Algorithms for Some Reservoir Engineering Problems"

Mary Wheeler, Rice University/University of Houston, US.

Parallel algorithms based on domain decomposition and operator splitting are presented for some problems in reservoir engineering. Theoretical and computational results are both discussed.

"An Additive Variant of the Schwarz Alternating Method for the Case of Many Subregions"

Olof Widlund, New York University.

In recent years, there has been a revival of interest in the Schwarz alternating method. Other domain decomposition algorithms, in particular, so-called iterative substructuring methods, have also been developed to solve elliptic finite element problems.

In this paper, we present an additive variant of the Schwarz method for elliptic problems, which shows great promise for parallel computers. By using ideas previously proven very useful for iterative substructuring methods, we are able to design a method which converges with a rate, which is independent of the mesh size as well as the number of subregion.

We note that, as is the case with other domain decomposition methods, a mechanism for global transportation of information is needed in order to obtain fast convergence if there are many subregions. In our algorithm this is accomplished by solving a global coarse problem in addition to the local problems typically associated with Schwarz' algorithm.

"Spectral Decomposition Methods for the Numerical Solution of Partial Differential Equations Using Vector and Parallel Processors"

David M. Young, the University of Texas, US.

The paper is concerned with the numerical solution of partial differential equations based on finite difference methods or on finite element methods. Iterative methods are used to solve the large, sparse linear systems of equations which arise. The emphasis is on the use of algorithms which are suitable for use on vector and parallel processors.

The focus is on methods which are based, not upon the decomposition of the physical domain, but rather on the decomposition of the eigenvalue spectrum of the coefficient matrix A or of a preconditioned matrix with the some basic iterative method such as the Jacobi method or the symmetric SOR method. These spectral decomposition methods are related to multigrid methods as well as to additive correction methods used by Adams (1985) and to multisplitting methods of O'Leary and White (1985).

They involve first splitting the initial residuals corresponding to several initial approximate solutions into several components each of which is primarily associated with a part of the eigenvalue spectrum. Each such component can then be treated independently in parallel and the results combined to minimize the norm of the error.

The process can be repeated. Numerical results illustrating the use of the procedure are given.

"Vector and Parallel Computing for Nonlinear Network Optimization"

Stavros A. Zenios, University of Pennsylvania, Philadelphia.

Nonlinear optimization models with network constraints are recurrent in applications in transportation, economics, statistics, operations research, and so on. We will discuss the design and implementation of parallel algorithms for this problem class.

First, a primal truncated Newton algorithm has been streamlined for the environment of the CRAY X-MP/48 using both vectorization and microtasking. The resultant algorithm is improved in performance by a factor of five over the scalar code.

Second, we design and implement a relaxation algorithm on a massively parallel Connection Machine, CM-1. One of the largest problem instances is solved in less than two seconds on the CM-1. The same problem takes 1.5 CPU minutes on a vector supercomputer and many hours (>8) on a minicomputer.

The algorithms will be introduced, followed by a discussion of the parallel implementations on the CM-1 and the CRAY X-MP/48. We will also present computational results with applications data and randomly generated problems.

"An Advanced Programing Environment for a Supercomputer"

Hans P. Zima, Bonn University, West Germany.

This paper discusses the programing environment that is currently being developed for the German supercomputer SUPRENUM. The SUPRENUM system is a loosely-coupled, massively parallel multiprocessing system whose overall design is oriented towards the solution of large-scale problems in numerical simulation and scientific supercomputing, with a particular emphasis on the multigrid method for the solution of large systems of partial differential equations.

The paper concentrates on two tools of the programing environment that are specifically tailored to support the kernel class of applications on a high level of abstraction, and at the same time achieve a high degree of runtime efficiency. These tools are the semiautomatic parallelization system SUPERB and the very high-level specification system SUSPENSE.

The objective of SUPERB is not only the parallelization of "dusty-deck" programs by transforming sequential FORTRAN 77 programs into parallel programs in SUPRENUM FORTRAN (a parallel superset of FORTRAN 77 supporting the features of the SUPRENUM architecture); in addition to this it will play a crucial role in the major program development paradigm of SUPRENUM which specifies a multistage process that starts with the writing of a sequential program, and proceeds to the final parallel SUPRENUM FORTRAN program in a number of steps that are associated with the step-by-step validation and verification of various properties of the program.

The specification system SUSPENSE enables programing on a very high, application-oriented level; it allows the formulation of partial differential equations and their solution methods in the language of numerical mathematics. The largely automatic process of transforming specifications into SUPRENUM FORTRAN programs is being supported by a knowledge base that is part of the system.

Student Scholarship Winners

"Efficient Parallel Programs Through Pipelined Block Algorithms, the QR Decomposition as an Example"

Christian Bischof, Cornell University, New York.

Block algorithms are effective for obtaining efficient and portable algorithms on computers with special vector processing hardware. We show that these advantages extend to parallel architectures. Combining block-oriented algorithms and pipelining, we obtain programs that have good processor utilization and load balancing properties and are easy to port across different architectures. We illustrate this technique with an algorithm for computing the QR factorization.

"An OR-Parallel Execution Model for Full Prolog"

Johannes Engels, Institut fur Informatik, Abteilung III, Rheinische Friedrich-Wilhelms-Universität, Bonn, West Germany.

This paper describes an OR-parallel execution model for Prolog. The goal is to execute full Prolog rather than a subset of Prolog in a fast way. Space optimization is not considered.

We want to break with the prejudice that "Many of the extensions of logic programing included in most Prolog systems are meaningful only in single-processor, sequential systems. Most notable are assert and retract... and the cut operation..."

"Parallel Compact Symmetric FFTs"

Van Emden Henson, University of Colorado, Boulder.

Since its introduction in 1965, the Fast Fourier Transform has become one of the most widely used algorithms of computational mathematics. Modern signal processing, spectroscopy, image processing, and techniques for solving partial differential equations all depend heavily on the FFT. Its enormous popularity is due largely to the fact that the FFT requires $O(N\log N)$ arithmetic operations to compute the transform of a complex vector of length N, instead of the $O(N)_2$ operations required to compute the transform as a matrix-vector product. The description "compact symmetric" refers to a family of FFT algorithms that uses minimal storage and arithmetic for data sequences possessing certain symmetries.

The first such algorithm, generally attributed to Edson, computes the transform of a real vector using half the storage and half the operations used by the original FFT. It has long been known that further savings are possible when the data has additional symmetries, but with the exception of one little-publicized algorithm by Gentleman, such transforms were performed by pre- and post-processing of data for use with conventional FFT's, which resulted in marginal savings.

In recent papers by Swarztrauber and Briggs, compact algorithms are developed for sequences with real, even, odd, quarter wave even, and quarter wave odd symmetries. These algorithms, like the original Cooley-Tukey FFT, work because the transform of a sequence can be formed by combining the transforms of two half-length sequences with very little work. By successively splitting the original N-vector, eventually N sequences of length one are produced, which are their own transforms. These are combined to form the transforms of sequences of length two, which in turn are combined to form the transforms of sequences of length four. After log$N$ passes through the data the transform of the original vector is produced. The new compact FFT's depend on the fact that when symmetric sequences are split the resulting subsequences inherit symmetries (For example, when an even sequence is split, one of the resulting sequences is even, while the other is quarter wave even). Performing the FFT then consists of modifying the Cooley-Tukey "combine" formulae to account for the changing symmetries of the partial sub-sequences. The transforms can then be performed in-place (without using an extra storage array). Implementing an algorithm requires a data structure that exploits the compactness allowed by the symmetry.

This paper shows how the compact symmetric algorithms can be implemented on parallel processing machines. Serial codes for the compact symmetric FFT's, not previously available, are implemented as a prelude to parallelizing the algorithms.

"Vectorizing the Multiple-Shooting Method for the Solution of Boundary-Value Problems and Optimal-Control Problems"

Martin Kiehl, Munich University of Technology, West Germany.

Many optimization problems in science and engineering can be often described by optimal-control problems, such as the control of the flight of an aircraft, the movement of a robot, or a chemical reaction. These problems become more complicated, since the solution has to consider constraints, so that, for example, technical and security limits or given tolerances are satisfied. This is the general case in realistic models.

Many of these problems can successfully be solved by the multiple-shooting method. It is therefore of great interest, to make this algorithm as fast as possible, and since vector computers are especially available in areas dealing with the above mentioned problems, the adaption of the algorithm to the new computer generation is impatiently awaited by various users.

As indicated by the name, vector computers were conceived for the fast solution of linear problems. But they can also be used for nonlinear problems, which can be solved by a sequence of linear problems. Unfortunately the multiple-shooting method is not of that type. Therefore it seems to be very difficult to take advantage of vector computers in this case. Nevertheless we will show, that it is still possible.

"The 3-D Linear Hierarchical Basis Preconditioner"

Maria Elizabeth G. Ong, University of Washington, Seattle.

The finite element discretization of a self-adjoint and positive definite problem in two dimensions using linear triangular elements and nodal basis functions generates a coefficient matrix A having a condition number $O(N)$, where N is the number of unknowns. Yserentant (1986) has shown that this can be improved to $O(\log\sqrt{N})^2)$ by using the hierarchical basis functions. This improvement can be viewed as preconditioning the linear system associated with the nodal basis functions, i.e., $A = S^T AS$, where A is the hierarchical basis coefficient matrix, A is the nodal basis coefficient matrix and S is a mesh-dependent matrix. This preconditioning can be implemented efficiently without forming the matrix S. A parallel implementation of this preconditioner in 2D applied to a Poisson problem with Dirichlet boundary conditions has been done by Adams and Ong (1987) on the Flexible-32 parallel processor at the NASA Langley Research Center, and by Greenbaum, Li, and Chao (1987) on the NYU Ultra-computer Prototype. These papers show that this preconditioner can be implemented more efficiently in parallel than ICCG. This fact, coupled with its logarithmic condition number and robustness makes the method very attractive.

In this paper, we study the performance of this method for 3-D problems. First, we extend Yserentant's results to three dimensions and show that the condition number is $O(N^{1/3})$ as opposed to $O(N^{2/3})$ for the nodal basis functions. The proof is constructed using linear tetrahedral elements. A refinement procedure is chosen such that a tetrahedron at level k is divided into eight equivolume tetrahedrons at level $k+1$. The number of nodes with j levels of refinement is given by $(2^j + 1)^3 = N$. Hierarchical basis functions are then defined at the nodes, and the discrete solution is expressed as a linear combination of these basis functions. With the aid of interpolating polynomials defined at each level of refinement, the solution can be decomposed into hierarchical basis components at each level. Using this procedure and an inequality we derived for a function defined on a sphere, we confirm Yserentant's conjecture that the condition number of the coefficient matrix A using hierarchical basis functions is bounded by $O(N^{1/3})$. We verify this result by comparing iteration counts to solve the linear system using the preconditioned conjugate gradient method.

Next, we discuss a strategy for implementing the 3-D hierarchical basis preconditioner on parallel processors with shared memory such as the NASA Langley's Flexible-32. For 2-D problems, multiplication by $A = S^T AS$ can be efficiently implemented and operations can be saved so that the work count of the preconditioned conjugate gradient method is $O(NlogN)$. The corresponding issue of the efficient implementation of multiplication by $A = S^T AS$ will be discussed for 3-D problems.

"A New Parallel Algorithm for LU Decomposition"

He Zhang, Department of Mathematics, Temple University, Philadelphia, Pennsylvania.

In this paper, we will develop a new parallel scheme for factoring a given matrix A into its lower (L) and upper (U) factored matrices. Based on this technique, a near-optimal parallel algorithm for LU decomposition is developed, where some of its important features are:

- Its design is insensitive to any number of processors, and its performance grows monotonically with them.
- It is especially good for large matrices, with dimensions large relatively to the number of processors in the system. In this case, it achieves the optimal speed-up, optimal efficiency, and very low communication complexity.
- It can be used in both distributed parallel computing environment and tightly coupled parallel computing system.
- This algorithm can be mapped onto any parallel architecture without any major programing difficulties or algorithmic changes.